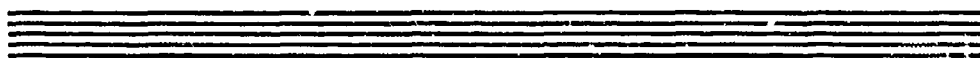




Key Practices of the Capability Maturity Model, Version 1.1



Mark C. Paulk
Charles V. Weber
Suzanne M. Garcia
Mary Beth Chrissis
Marilyn Bush

DTIC
SELECTE
APR 27 1993
S E D

DISTRIBUTION STATEMENT
Approved for public release
Distribution Unlimited

93-08747



49988

February 1993
CMU/SEI-93-TR-25
ESC-TR-93-178

Key Practices of the Capability Maturity Model, Version 1.1



Mark C. Paulk
Charles V. Weber
Suzanne M. Garcia
Mary Beth Chrissis
Marilyn Bush

Approved for public release
Distribution unlimited

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

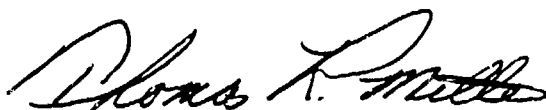
SEI Joint Program Office
ESC/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER



Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Copyright © 1993 by Carnegie Mellon University.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.



Carnegie Mellon University
Software Engineering Institute

Permission to reproduce, in whole or in part, the volume of materials released by the Software Engineering Institute under the title *Capability Maturity Model for Software* is granted under the following conditions:

1. This letter must be reproduced with each copy.
2. All copies must include the copyright notice.
3. The materials are not to be used for commercial gain.
4. The materials are not to be distributed beyond your organization. Refer such requests to the SEI.
5. The materials are to be used in a manner consistent with the framework and methodology advanced by the SEI.
6. Carnegie Mellon University and the Software Engineering Institute are not to be construed as responsible for the results of analyses conducted as a result of this permission. In other words, neither CMU nor the SEI is to be held liable for your use of this material.

Sincerely,

Purvis M. Jackson
Sr. Editor and Director
SEI Information Management

Accession For	
NTIS	CRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced <input type="checkbox"/>	
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

DATE

Table of Contents

Acknowledgments.....	O-vii
----------------------	-------

Overview

1	Introducing the Key Practices Document.....	O-1
1.1	To the Reader.....	O-1
1.2	Relationship of this Document to Other Documents.....	O-3
1.3	Organization of this Document.....	O-4
1.4	Expected Use of this Document.....	O-5
2	Overview of the Capability Maturity Model.....	O-7
2.1	Introducing the Capability Maturity Model.....	O-7
2.2	Sources of the CMM.....	O-7
2.3	Structure of the CMM.....	O-8
2.4.1	Level 1 - The Initial Level.....	O-13
2.4.2	Level 2 - The Repeatable Level.....	O-14
2.4.3	Level 3 - The Defined Level.....	O-15
2.4.4	Level 4 - The Managed Level.....	O-16
2.4.5	Level 5 - The Optimizing Level.....	O-17
2.5	The Key Process Areas of the CMM.....	O-18
2.6	The Key Practices.....	O-26
2.7	Goals.....	O-27
2.8	Common Features.....	O-27
3	Using the Key Practice Pages.....	O-31
4	Interpreting the CMM.....	O-35
4.1	Interpreting the Key Practices.....	O-35
4.2	Interpreting the Common Features.....	O-35
4.2.1	Commitment to Perform.....	O-36
4.2.2	Ability to Perform.....	O-37
4.2.3	Activities Performed.....	O-39
4.2.4	Measurement and Analysis.....	O-47
4.2.5	Verifying Implementation.....	O-47
4.3	Interpreting Software Process Definition.....	O-50
4.3.1	Process Definition Concepts.....	O-51
4.3.2	Concepts Related to the Organization's Software Process Assets.....	O-53
4.3.3	Concepts Related to the Project's Defined Software Process.....	O-58

Table of Contents

4.3.4	Relationship Between the Project's Defined Software Process and the Software Development Plan.....	O-61
4.3.5	Life Cycles and the CMM.....	O-61
4.3.6	Technology and the CMM	O-62
4.3.7	Documentation and the CMM.....	O-62
4.3.8	Collection and Analysis of Process Data	O-63
4.4	Organizational Structure and Roles	O-64
4.4.1	Organizational Roles.....	O-64
4.4.2	Organizational Structure.....	O-69
4.4.3	Independence and Organizational Structure.....	O-73
4.5	Applying Professional Judgment	O-74
 Level 2 Key Practices		
	Requirements Management.....	L2-1
	Software Project Planning.....	L2-11
	Software Project Tracking and Oversight.....	L2-29
	Software Subcontract Management.....	L2-43
	Software Quality Assurance	L2-59
	Software Configuration Management	L2-71
 Level 3 Key Practices		
	Organization Process Focus	L3-1
	Organization Process Definition.....	L3-11
	Training Program	L3-25
	Integrated Software Management.....	L3-37
	Software Product Engineering	L3-59
	Intergroup Coordination.....	L3-83
	Peer Reviews	L3-93
 Level 4 Key Practices		
	Quantitative Process Management.....	L4-1
	Software Quality Management.....	L4-19
 Level 5 Key Practices		
	Defect Prevention	L5-1
	Technology Change Management.....	L5-17
	Process Change Management.....	L5-31

Table of Contents

Appendices

Appendix A: References..	A-1
Appendix B: Glossary of Terms	A-3
Appendix C: Abridged Version of the Key Practices	A-27
Appendix D: Change History	A-67
Appendix E: Index	A-73

Table of Contents

List of Figures

Figure 2.1	The Structure of the Capability Maturity Model.....	O-9
Figure 2.2	The Five Levels of Software Process Maturity	O-13
Figure 2.3	The Key Process Areas by Maturity Level.....	O-19
Figure 2.4	The Key Process Areas Assigned to Process Categories	O-25
Figure 3.1	Example of Key Practice Statements	O-33
Figure 4.1	Conceptual Software Process Framework Used in the CMM.....	O-52

List of Figures

Acknowledgments

The SEI would like to thank the many people who were involved in the development of this document. The SEI has been developing and revising the Capability Maturity Model for Software (CMM) since 1988 based on the knowledge acquired from software process assessments, software capability evaluations, and feedback from both industry and government. This document was produced as part of the revision of CMM Version 1.0, which was released in August, 1991.

The conceptual framework for the capability maturity model was developed by Watts Humphrey. He has advised us as we refined the model and evolved the practices.

This document has taken many drafts to evolve into the current product. We would like to thank the individuals who have worked many hours developing ideas and formulating the practices for this model. In particular, we acknowledge the contributions of Charlie Weber, Mark Paulk, Cynthia Wise, Jim Withey, Mary Beth Chrissis, Suzanne Garcia, and Marilyn Bush.

Many other individuals have helped with their comments, criticisms, and suggestions. They include Ed Averill, Judy Bamberger, Anita Carleton, Bill Curtis, Susan Dart, Lionel Deimel, Peter Feiler, Julia Gale, Jim Hart, Ron Higuera, Purvis Jackson, Tim Kasse, David Kitson, Mike Konrad, Peter Malpass, Mark Manduke, Judah Mogilensky, Warren Moseley, Jim Over, George Pandelios, Bob Park, Jeff Perdue, Dick Phillips, Mike Rissman, Jim Rozum, and Christer von Schantz.

Special thanks go to the members of the CMM Correspondence Group who contributed their time and effort to reviewing drafts of the CMM and providing insightful comments and recommendations. We would like to thank especially those who took time to participate in the March 1990 and April 1992 CMM Workshops, as well as those who took time to meet with us on other occasions to provide comments and recommendations. This effort could not have been accomplished without these practitioners lending their expertise to refine the model.

We would also like to thank the members of the CMM Advisory Board who helped guide us in our efforts. In addition to providing technical insights,

Acknowledgments

they helped focus our effort and worked with us to evaluate and plan our actions to address the many comments received from industry and government reviewers. The current members are Constance Ahara, Kelley Butler, Bill Curtis, Conrad Czaplicki, Raymond Dion, Judah Mogilensky, Martin Owens, Mark Paulk, Sue Stetak, Charlie Weber, and Ronald Willis. Former members who worked with us on CMM v1.0 include Harry Carl, Jim Hess, Jerry Pixton, and Jim Withey.

Lastly, we would like to thank the many individuals who helped produce this document. We would like to thank the members of the Joint Program Office who expedited the approval process. We would also like to thank Ginny Redish and Renee Dutkowski from the American Institutes for Research for their help with editing and designing the document. We very much appreciate the efforts of Carolyn Tady, David White, and Debbie Punjack for their administrative support, and also Mary Beth Chrissis, Suzanne Couturiaux, and Mike Konrad who did the editing, formatting, and whatever else was necessary to get the document produced.

Capability Maturity Model

Overview of the Key Practices

This part presents an overview of the key practices that make up the Capability Maturity Model. After introducing the key practices document, it briefly describes the Capability Maturity Model. Next, it offers some advice for interpreting and using the key practice statements.

	<i>Overview</i>
Introducing the Key Practices Document	O-1
Overview of the Capability Maturity Model	O-7
Using the Key Practice Pages	O-31
Interpreting the CMM	O-35

1 Introducing the Key Practices Document

1.1 To the Reader

Developing reliable and usable software that is delivered on time and within budget is a difficult endeavor for many organizations. Products that are late, over budget, or that don't work as expected also cause problems for the organization's customers. As software projects continue to increase in size and importance, these problems become magnified. These problems can be overcome through a focused and sustained effort at building a process infrastructure of effective software engineering and management practices.

To build this process infrastructure, organizations producing software need ways to appraise their ability to perform their software process successfully. They also need guidance to improve their process capability. Customers, such as the Department of Defense (DoD), need ways to evaluate more effectively an organization's capability to perform successfully on software engineering contracts. Prime contractors need ways to evaluate the capability of potential subcontractors.

To help organizations and customers like the DoD and prime contractors, the Software Engineering Institute (SEI) has developed the Capability Maturity Model for Software (CMM), that delineates the characteristics of a mature, capable software process. The progression from an immature, unrepeatable software process to a mature, well-managed software process also is described in terms of maturity levels in the model.

The CMM can be used for:

- ❑ software process improvement, in which an organization plans, develops, and implements changes to its software process;

Introducing the Key Practices Document

- ❑ software process assessments, in which a trained team of software professionals determines the state of an organization's current software process, determines the high-priority software process-related issues facing an organization, and obtains the organizational support for software process improvement; and
- ❑ software capability evaluations, in which a trained team of professionals identifies contractors who are qualified to perform the software work or monitors the state of the software process used on an existing software effort.

This document describes the key practices that correspond to each maturity level in the CMM. It is an elaboration of what is meant by maturity at each level of the CMM and a guide that can be used for software process improvement, software process assessments, and software capability evaluations.

The key practices of the CMM are expressed in terms of what is expected to be the normal practices of organizations that work on large, government contracts. In any context in which the CMM is applied, a reasonable interpretation of how the practices would be applied should be used. Guidelines on interpreting the CMM are contained in Chapter 4 of this document. The CMM must be appropriately interpreted when the business environment of the organization differs significantly from that of a large contracting organization. The role of professional judgement in making informed use of the CMM must be recognized.

This document can be used in several ways:

- ❑ by anyone wanting to understand the key practices that are part of effective processes for developing or maintaining software,

- ☐ by anyone wanting to identify the key practices that are needed to achieve the next maturity level in the CMM,
- ☐ by organizations wanting to understand and improve their capability to develop software effectively,
- ☐ by acquisition organizations or prime contractors wanting to identify the risks of having a particular organization perform the work of a contract,
- ☐ by the SEI as the basis for developing process products, such as the maturity questionnaire, and
- ☐ by instructors preparing teams to perform software process assessments or software capability evaluations.

1.2 Relationship of this Document to Other Documents

The two documents that provided the initial foundation for the CMM are:

- ☐ "Characterizing the Software Process" [Humphrey88], and
- ☐ *Managing the Software Process* [Humphrey89].

Version 1.0 of the CMM was released in August of 1991 in two technical reports:

- ☐ "Capability Maturity Model for Software" [Paulk91], and
- ☐ "Key Practices of the Capability Maturity Model" [Weber91].

This initial release of the CMM was revised during 1992. To understand and use the current version of the CMM, two documents are needed:

Introducing the Key Practices Document

- ❑ "Capability Maturity Model for Software, Version 1.1" [Paulk93a], and
- ❑ this document, "Key Practices of the Capability Maturity Model, Version 1.1" [Paulk93b].

"Capability Maturity Model for Software, Version 1.1" contains an introduction to the model, descriptions of the five maturity levels, an operational definition of the CMM and its structure, a discussion of how organizations can use the maturity model, and some remarks on the future directions of the CMM.

"Key Practices of the Capability Maturity Model, Version 1.1," contains the key practices that correspond to the key process areas at each maturity level of the CMM and information to help interpret the key practices.

The maturity questionnaire and other process products are derived from the key practices of the Capability Maturity Model. Other SEI process products that support software process improvement, software process assessment, and software capability evaluation include training courses, handbooks, and site visit guides.

1.3 Organization of this Document

This first chapter gives an overview of the CMM and of this document. In the next three chapters of the overview are:

- ❑ an overview of the CMM and its constituent parts,
- ❑ a description of how to use the format of the key practices, and
- ❑ a description of ways to use and interpret the key practices.

Following the overview, the key practices for the key process areas of the CMM are described. For those who want to get a quick sense of the key practices, without the rigor that is needed in applying them, an abridgment of the key practices is provided in Appendix C.

In the appendices are a list of the references cited in this document, a glossary of terms used in this document, an abridgment of the key practices, the change history for this document, and an index of terms contained in this document.

1.4 Expected Use of this Document

If you are not familiar with the CMM, you should first read the paper, "Capability Maturity Model for Software, Version 1.1" [Paulk93a] and all four chapters in this overview before trying to use the key practices.

If you are already familiar with the CMM and how it is structured, you may want to go directly to the fourth chapter for advice on how to interpret the key practices.

Introducing the Key Practices Document

2 Overview of the Capability Maturity Model

2.1 Introducing the Capability Maturity Model

The Capability Maturity Model for Software (CMM) is a framework that describes the key elements of an effective software process. The CMM describes an evolutionary improvement path from an ad hoc, immature process to a mature, disciplined process.

The CMM covers practices for planning, engineering, and managing, software development and maintenance. When followed, these key practices improve the ability of organizations to meet goals for cost, schedule, functionality, and product quality.

The CMM establishes a yardstick against which it is possible to judge, in a repeatable way, the maturity of an organization's software process and compare it to the state of the practice of the industry [Kitson92]. The CMM can also be used by an organization to plan improvements to its software process.

2.2 Sources of the CMM

The Software Engineering Institute (SEI) developed an initial version of a maturity model and maturity questionnaire at the request of the government and with the assistance of the MITRE Corporation. Throughout the development of the model and the questionnaire, the SEI has paid attention to advice from practitioners who are involved in developing and improving software processes. Our objective has been to provide a model that:

- ☐ is based on actual practices;

Overview of the Capability Maturity Model

- ☐ reflects the best of the state of the practice;
- ☐ reflects the needs of individuals performing software process improvement, software process assessments, or software capability evaluations;
- ☐ is documented; and
- ☐ is publicly available.

Additional knowledge and insight into software process maturity has been gained since the earlier versions of the maturity model. This insight has been gained by:

- ☐ studying non-software organizations,
- ☐ performing and observing software process assessments and software capability evaluations,
- ☐ soliciting and analyzing change requests to the model,
- ☐ participating in meetings and workshops with industry and government representatives, and
- ☐ soliciting feedback from industry and government reviewers.

Using this additional knowledge, the Capability Maturity Model and its practices have been revised, creating CMM v1.1.

2.3 Structure of the CMM

The CMM is composed of five maturity levels. With the exception of Level 1, each maturity level is composed of several key process areas. Each key process area is organized into five sections called common features. The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area. This structure of the CMM is illustrated in Figure 2.1.

Overview of the Capability Maturity Model

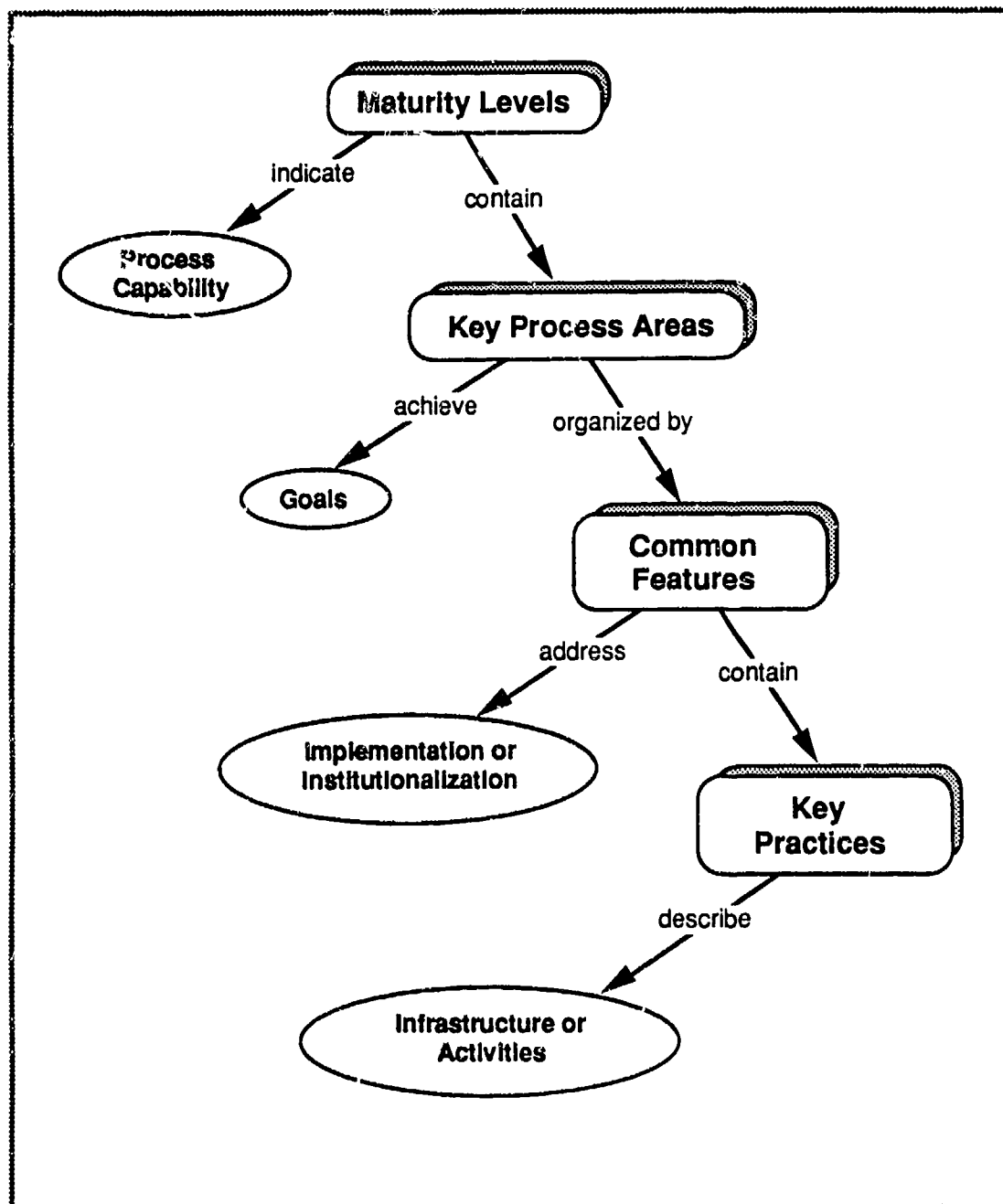


Figure 2.1 The Structure of the Capability Maturity Model

Overview of the Capability Maturity Model

The components of the CMM include:

- Maturity levels* A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. The five maturity levels provide the top-level structure of the CMM.
- Process capability* Software process capability describes the range of expected results that can be achieved by following a software process. The software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes.
- Key process areas* Each maturity level is composed of key process areas. Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability at that maturity level. The key process areas have been defined to reside at a single maturity level. For example, one of the key process areas for Level 2 is Software Project Planning.

Overview of the Capability Maturity Model

Goals

The goals summarize the key practices of a key process area and can be used to determine whether an organization or project has effectively implemented the key process area. The goals signify the scope, boundaries, and intent of each key process area.

An example of a goal from the Software Project Planning key process area is "Software estimates are documented for use in planning and tracking the software project." See "Capability Maturity Model for Software, Version 1.1" [Paulk93a] and Section 4.5, Applying Professional Judgment, of this document for more information on interpreting the goals.

Common features

The key practices are divided among five Common Features sections: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting.

The Activities Performed common feature describes implementation activities. The other four common features describe the institutionalization factors, which make a process part of the organizational culture.

Overview of the Capability Maturity Model

Key practices

Each key process area is described in terms of key practices that, when implemented, help to satisfy the goals of that key process area. The key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

For example, one of the practices from the Software Project Planning key process area is "The project's software development plan is developed according to a documented procedure."

2.4 Definition of the CMM Maturity Levels

As organizations establish and improve the software processes by which they develop and maintain their software work products, they progress through levels of maturity. Figure 2.2 shows the five maturity levels of the CMM.

Each maturity level provides a layer in the foundation for continuous process improvement. Each key process area comprises a set of goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity model institutionalizes a different component in the software process, resulting in an overall increase in the process capability of the organization.

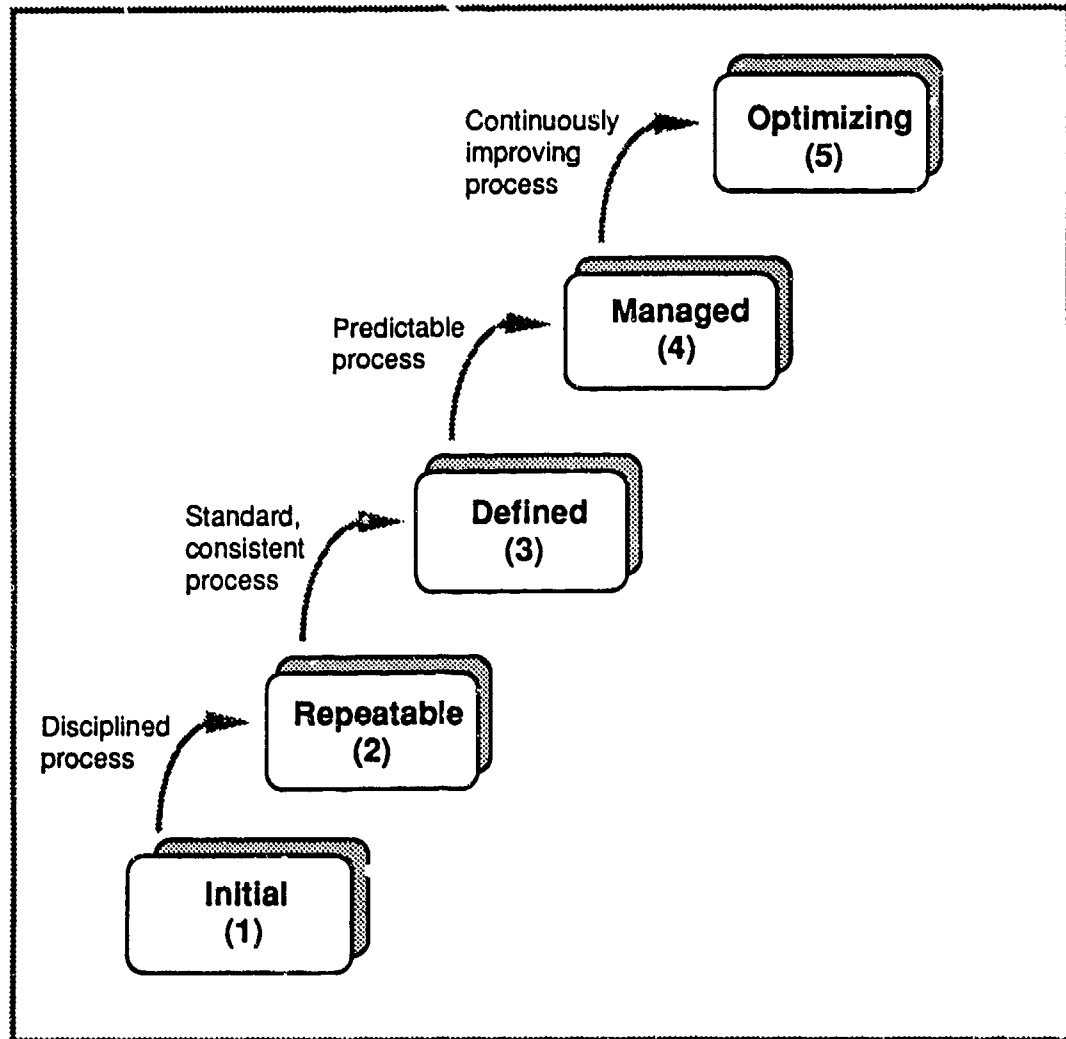


Figure 2.2 The Five Levels of Software Process Maturity

2.4.1 Level 1 - The Initial Level

At the Initial Level, the organization typically does not provide a stable environment for developing and maintaining software. When an organization lacks sound management practices, the benefits of good

Overview of the Capability Maturity Model

software engineering practices are undermined by ineffective planning and reaction-driven commitment systems.

During a crisis, projects typically abandon planned procedures and revert to coding and testing. Success depends entirely on having an exceptional manager and a seasoned and effective software team. Occasionally, capable and forceful software managers can withstand the pressures to take shortcuts in the software process; but when they leave the project, their stabilizing influence leaves with them. Even a strong engineering process cannot overcome the instability created by the absence of sound management practices.

The software process capability of Level 1 organizations is unpredictable because the software process is constantly changed or modified as the work progresses (i.e., the process is ad hoc). Schedules, budgets, functionality, and product quality are generally unpredictable. Performance depends on the capabilities of individuals and varies with their innate skills, knowledge, and motivations. There are few stable software processes in evidence, and performance can be predicted only by individual rather than organizational capability.

2.4.2 Level 2 - The Repeatable Level

At the Repeatable Level, policies for managing a software project and procedures to implement those policies are established. Planning and managing new projects is based on experience with similar projects. An objective in achieving Level 2 is to institutionalize effective management processes for software projects, which allow organizations to repeat successful practices developed on earlier projects, although the specific processes implemented by the projects may differ. An effective process can be characterized as practiced, documented, enforced, trained, measured, and able to improve.

Overview of the Capability Maturity Model

Projects in Level 2 organizations have installed basic software management controls. Realistic project commitments are based on the results observed on previous projects and on the requirements of the current project. The software managers for a project track software costs, schedules, and functionality; problems in meeting commitments are identified when they arise. Software requirements and the work products developed to satisfy them are baselined, and their integrity is controlled. Software project standards are defined, and the organization ensures they are faithfully followed. The software project works with its subcontractors, if any, to establish a strong customer-supplier relationship.

The software process capability of Level 2 organizations can be summarized as disciplined because planning and tracking of the software project is stable and earlier successes can be repeated. The project's process is under the effective control of a project management system, following realistic plans based on the performance of previous projects.

2.4.3 Level 3 - The Defined Level

At the Defined Level, the standard process for developing and maintaining software across the organization is documented, including both software engineering and management processes, and these processes are integrated into a coherent whole. This standard process is referred to throughout the CMM as the organization's standard software process. Processes established at Level 3 are used (and changed, as appropriate) to help the software managers and technical staff perform more effectively. The organization exploits effective software engineering practices when standardizing its software processes. There is a group that is responsible for the organization's software process activities, e.g., a software engineering process group, or SEPG [Fowler90]. An organization-wide training program is implemented to ensure that the staff and managers have the knowledge and skills required to fulfill their assigned roles.

Overview of the Capability Maturity Model

Projects tailor the organization's standard software process to develop their own defined software process, which accounts for the unique characteristics of the project. This tailored process is referred to in the CMM as the project's defined software process. A defined software process contains a coherent, integrated set of well-defined software engineering and management processes. A well-defined process can be characterized as including readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and completion criteria. Because the software process is well defined, management has good insight into technical progress on all projects.

The software process capability of Level 3 organizations can be summarized as standard and consistent because both software engineering and management activities are stable and repeatable. Within established product lines, cost, schedule, and functionality are under control, and software quality is tracked. This process capability is based on a common, organization-wide understanding of the activities, roles, and responsibilities in a defined software process.

2.4.4 Level 4 - The Managed Level

At the Managed Level, the organization sets quantitative quality goals for both software products and processes. Productivity and quality are measured for important software process activities across all projects as part of an organizational measurement program. An organization-wide software process database is used to collect and analyze the data available from the projects' defined software processes. Software processes are instrumented with well-defined and consistent measurements at Level 4. These measurements establish the quantitative foundation for evaluating the projects' software processes and products.

Overview of the Capability Maturity Model

Projects achieve control over their products and processes by narrowing the variation in their process performance to fall within acceptable quantitative boundaries. Meaningful variations in process performance can be distinguished from random variation (noise), particularly within established product lines. The risks involved in moving up the learning curve of a new application domain are known and carefully managed.

The software process capability of Level 4 organizations can be summarized as predictable because the process is measured and operates within measurable limits. This level of process capability allows an organization to predict trends in process and product quality within the quantitative bounds of these limits. When these limits are exceeded, action is taken to correct the situation. Software products are of predictably high quality.

2.4.5 Level 5 - The Optimizing Level

At the Optimizing Level, the entire organization is focused on continuous process improvement. The organization has the means to identify weaknesses and strengthen the process proactively, with the goal of preventing the occurrence of defects. Data on the effectiveness of the software process is used to perform cost benefit analyses of new technologies and proposed changes to the organization's software process. Innovations that exploit the best software engineering practices are identified and transferred throughout the organization.

Software project teams in Level 5 organizations analyze defects to determine their causes. Software processes are evaluated to prevent known types of defects from recurring, and lessons learned are disseminated to other projects.

The software process capability of Level 5 organizations can be characterized as continuously improving because Level 5 organizations are continuously

Overview of the Capability Maturity Model

striving to improve the range of their process capability, thereby improving the process performance of their projects. Improvement occurs both by incremental advancements in the existing process and by innovations using new technologies and methods.

2.5 The Key Process Areas of the CMM

Figure 2.3 lists the key process areas for each maturity level in the CMM. Each *key process area* identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. The key process areas have been defined to reside at a single maturity level. The key process areas are building blocks that indicate the areas an organization should focus on to improve its software process. Key process areas identify the issues that must be addressed to achieve a maturity level.

Overview of the Capability Maturity Model

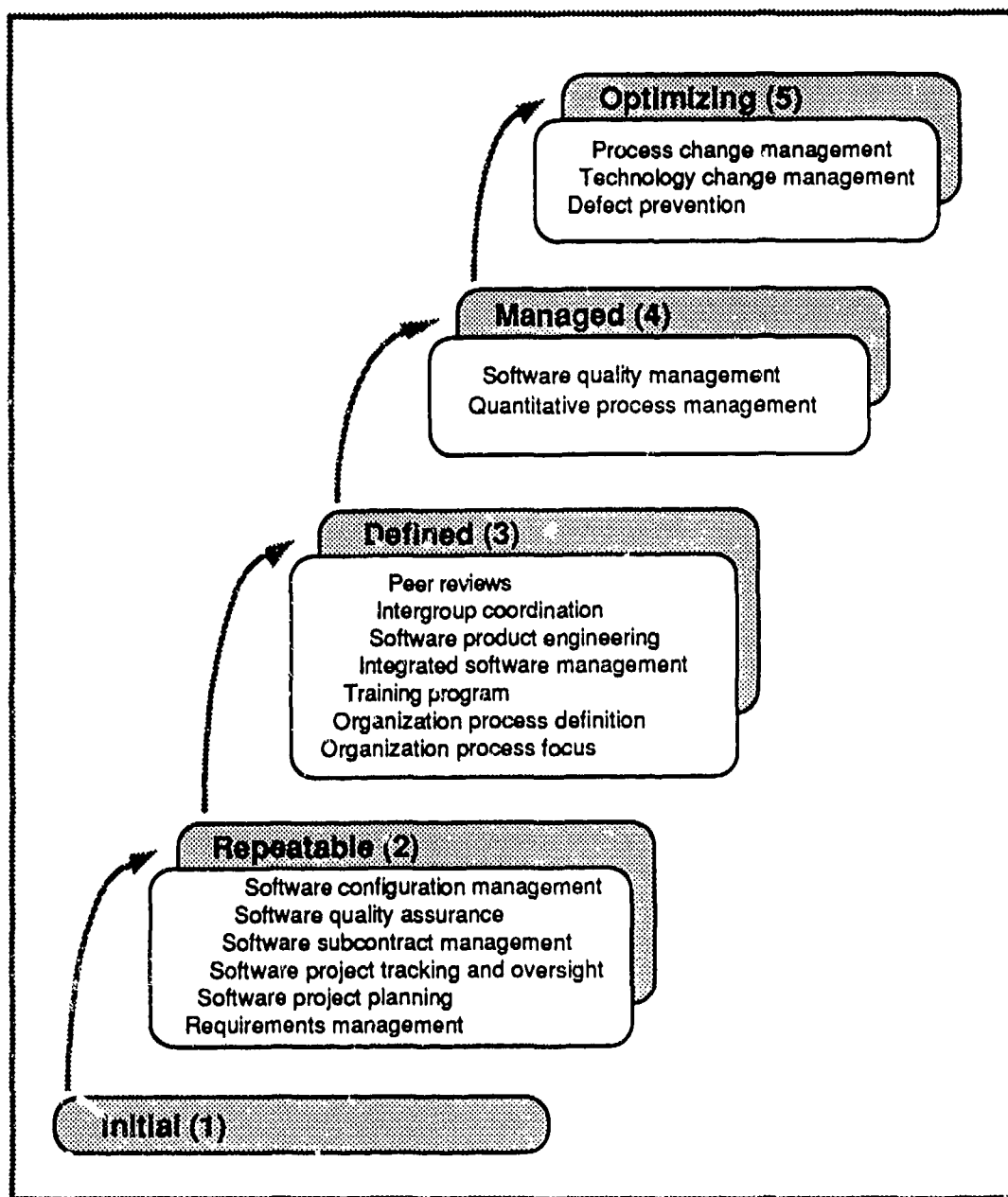


Figure 2.3 The Key Process Areas by Maturity Level

Overview of the Capability Maturity Model

The key process areas at Level 2 focus on the software project's concerns related to establishing basic project management controls. Descriptions of each of the key process areas for Level 2 are given below:

- ❑ The purpose of Requirements Management is to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project. This agreement with the customer is the basis for planning (as described in Software Project Planning) and managing (as described in Software Project Tracking and Oversight) the software project. Control of the relationship with the customer depends on following an effective change control process (as described in Software Configuration Management).
- ❑ The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. These plans are the necessary foundation for managing the software project (as described in Software Project Tracking and Oversight). Without realistic plans, effective project management cannot be implemented.
- ❑ The purpose of Software Project Tracking and Oversight is to establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.
- ❑ The purpose of Software Subcontract Management is to select qualified software subcontractors and manage them effectively. It combines the concerns of Requirements Management, Software Project Planning, and Software Project Tracking and Oversight for basic management control, along with necessary coordination of Software Quality Assurance and Software Configuration Management, and applies this control to the subcontractor as appropriate.

Overview of the Capability Maturity Model

- ❑ The purpose of Software Quality Assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built. Software Quality Assurance is an integral part of most software engineering and management processes.
- ❑ The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle. Software Configuration Management is an integral part of most software engineering and management processes.

The key process areas at Level 3 address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects. Descriptions of each of the key process areas for Level 3 are given below:

- ❑ The purpose of Organization Process Focus is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability. The primary result of the Organization Process Focus activities is a set of software process assets, which are described in Organization Process Definition. These assets are used by the software projects, as is described in Integrated Software Management.
- ❑ The purpose of Organization Process Definition is to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization. These assets provide a stable foundation that can be institutionalized via mechanisms such as training, which is described in Training Program.

Overview of the Capability Maturity Model

- ❑ The purpose of Training Program is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently. Training is an organizational responsibility, but the software projects should identify their needed skills and provide the necessary training when the project's needs are unique.
- ❑ The purpose of Integrated Software Management is to integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets, which are described in Organization Process Definition. This tailoring is based on the business environment and technical needs of the project, as described in Software Product Engineering. Integrated Software Management evolves from Software Project Planning and Software Project Tracking and Oversight at Level 2.
- ❑ The purpose of Software Product Engineering is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently. Software Product Engineering describes the technical activities of the project, e.g., requirements analysis, design, code, and test.
- ❑ The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently. Intergroup Coordination is the interdisciplinary aspect of Integrated Software Management that extends beyond software engineering; not only should the software process be integrated, but the software engineering group's interactions with other groups must be coordinated and controlled.
- ❑ The purpose of Peer Reviews is to remove defects from the software work products early and efficiently. An important corollary effect is to

Overview of the Capability Maturity Model

develop a better understanding of the software work products and of the defects that can be prevented. The peer review is an important and effective engineering method that is called out in Software Product Engineering and that can be implemented via Fagan-style inspections [Fagan86], structured walkthroughs, or a number of other collegial review methods [Freedman90].

The key process areas at Level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built. The two key process areas at this level, Quantitative Process Management and Software Quality Management, are highly interdependent, as is described below:

- The purpose of Quantitative Process Management is to control the process performance of the software project quantitatively. Software process performance represents the actual results achieved from following a software process. The focus is on identifying special causes of variation within a measurably stable process and correcting, as appropriate, the circumstances that drove the transient variation to occur. Quantitative Process Management adds a comprehensive measurement program to the practices of Organization Process Definition, Integrated Software Management, Intergroup Coordination, and Peer Reviews.
- The purpose of Software Quality Management is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals. Software Quality Management applies a comprehensive measurement program to the software work products described in Software Product Engineering.

The key process areas at Level 5 cover the issues that both the organization and the projects must address to implement continuous and measurable software process improvement. Descriptions of each of the key process areas for Level 5 are given below:

Overview of the Capability Maturity Model

- ❑ The purpose of Defect Prevention is to identify the causes of defects and prevent them from recurring. The software project analyzes defects, identifies their causes, and changes its defined software process, as is described in Integrated Software Management. Process changes of general value are transitioned to other software projects, as is described in Process Change Management.
- ❑ The purpose of Technology Change Management is to identify beneficial new technologies (i.e., tools, methods, and processes) and transfer them into the organization in an orderly manner, as is described in Process Change Management. The focus of Technology Change Management is on performing innovation efficiently in an ever-changing world.
- ❑ The purpose of Process Change Management is to continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development. Process Change Management takes the incremental improvements of Defect Prevention and the innovative improvements of Technology Change Management and makes them available to the entire organization.

By definition, key process areas are expressed at a single maturity level. There are, however, relationships between the key process areas, and improvements in a specific management or technical area need not be restricted to a single key process area. Figure 2.4 illustrates these relationships. Organizations may work on higher level key process areas before they have achieved lower level maturity levels, and attention must continue to be focused on lower level key process areas even when key process areas at higher maturity levels have been achieved.

Overview of the Capability Maturity Model

Processes Categories	Management <i>Software project planning, management, etc.</i>	Organizational <i>Senior management review, etc.</i>	Engineering <i>Requirements analysis, design, code, test, etc.</i>
Levels			
5 Optimizing		Technology Change Management Process Change Management	Defect Prevention
4 Managed	Quantitative Process Management		Software Quality Management
3 Defined	Integrated Software Management Intergroup Coordination	Organization Process Focus Organization Process Definition Training Program	Software Product Engineering Peer Reviews
2 Repeatable	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management		
1 Initial	Ad Hoc Processes		

Figure 2.4 The Key Process Areas Assigned to Process Categories

Overview of the Capability Maturity Model

The key process areas are categorized in Figure 2.4 into three broad categories: Management, Organizational, and Engineering processes. The Management process category contains the project management activities as they evolve from planning and tracking at Level 2, to managing according to a defined software process at Level 3, to quantitative management at Level 4, to innovative management in a constantly changing environment at Level 5. The Organizational process category contains the cross-project responsibilities as the organization matures, beginning with a focus on process issues at Level 3, continuing to a quantitative understanding of the process at Level 4, and culminating with the management of change in an environment of continuous process improvement at Level 5. The Engineering process category contains the technical activities, such as requirements analysis, design, code, and test, which are performed at all levels, but that evolve toward an engineering discipline at Level 3, statistical process control at Level 4, and continuous measured improvement at Level 5.

Note that at Levels 4 and 5 there are key process areas that span these process categories. This helps identify potential new key process areas for CMM v2 as Levels 4 and 5 become better understood.

2.6 The Key Practices

Each key process area is described in terms of the key practices that contribute to satisfying its goals. The key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

Each key practice consists of a single sentence, often followed by a more detailed description, which may include examples and elaboration. These key practices, also referred to as the top-level key practices, state the fundamental policies, procedures, and activities for the key process area.

Overview of the Capability Maturity Model

The components of the detailed description are frequently referred to as subpractices.

The key practices describe "what" is to be done, but they should not be interpreted as mandating "how" the goals should be achieved. Alternative practices may accomplish the goals of the key process area. The key practices should be interpreted rationally to judge whether the goals of the key process area are effectively, although perhaps differently, achieved.

2.7 Goals

The goals summarize the key practices of a key process area and can be used to determine whether an organization or project has effectively implemented the key process area. The goals signify the scope, boundaries, and intent of each key process area. In adapting the key practices of a key process area to a specific project situation, the goals can be used to determine whether the adaptation is a reasonable rendering of the practices. Similarly, when assessing or evaluating alternative ways to implement a key process area, the goals can be used to determine if the alternatives satisfy the intent of the key process area. Please refer to "Capability Maturity Model for Software, Version 1.1" [Paulk93a] and Section 4.5, Applying Professional Judgment, of this document for more information on interpreting the goals in an organization.

2.8 Common Features

The key practices in each key process area are organized by a set of common features. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting. The common features also group and order the key practices in a sequence helpful for organizations using them. The five common features are listed below:

Overview of the Capability Maturity Model

Commitment to Perform

Commitment to Perform describes the actions the organization must take to ensure that the process is established and will endure. Commitment to Perform typically involves establishing organizational policies and senior management sponsorship.

Ability to Perform

Ability to Perform describes the preconditions that must exist in the project or organization to implement the software process competently. Ability to Perform typically involves resources, organizational structures, and training.

Activities Performed

Activities Performed describes the roles and procedures necessary to implement a key process area. Activities Performed typically involve establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.

Measurement and Analysis

Measurement and Analysis describes the need to measure the process and analyze the measurements. Measurement and Analysis typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed.

Verifying Implementation

Verifying Implementation describes the steps to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

Overview of the Capability Maturity Model

The practices in the common feature Activities Performed describe what must be implemented to establish a process capability. The other practices, taken as a whole, form the basis by which an organization can institutionalize the practices described in the Activities Performed common feature. The Activities Performed by projects or the organization provide the largest category of key practices because they describe the actual implementation of the key process area. Key practices under the other common features are equally important, however, for they address what must be done to support and institutionalize the key process area.

Overview of the Capability Maturity Model

3 Using the Key Practice Pages

The key practices are grouped by maturity level, and each maturity level is separated by a tab page. The tab page includes a description of the maturity level, a list of the key process areas for that maturity level, and the page number where each key process area begins.

Each key process area contains:

- ☐ a brief description of the key process area,
- ☐ the goals for the key process area, and
- ☐ the key practices.

The key practices themselves are grouped into the five common features (Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation) and are presented in a hierarchical format, as shown in Figure 3.1, an example page from the key practices. The key practices include:

Key practices

The key practices, also known as top-level key practices, state the fundamental policies, procedures, and activities for the key process area. They are identified in bold and are numbered within each common feature. For example, the first key practice in the common feature of Activities Performed is identified as Activity 1.

Subpractices

Subpractices, also known as subordinate key practices, are listed beneath the top-level key practices and describe what one would expect to find implemented for the top-level key practice. The subpractices can be used to help determine whether or not the key practices are implemented satisfactorily.

Using the Key Practice Pages

Supplementary information

Supplementary information is boxed following the key practices. The supplementary information includes examples, elaborations, and references to other key process areas.

When the subpractices or the supplementary information underneath a key practice extends to another page, the number of the key practice is shown in parentheses at the start of the new page to indicate that the information on that page is a continuation of the key practice on the previous page.

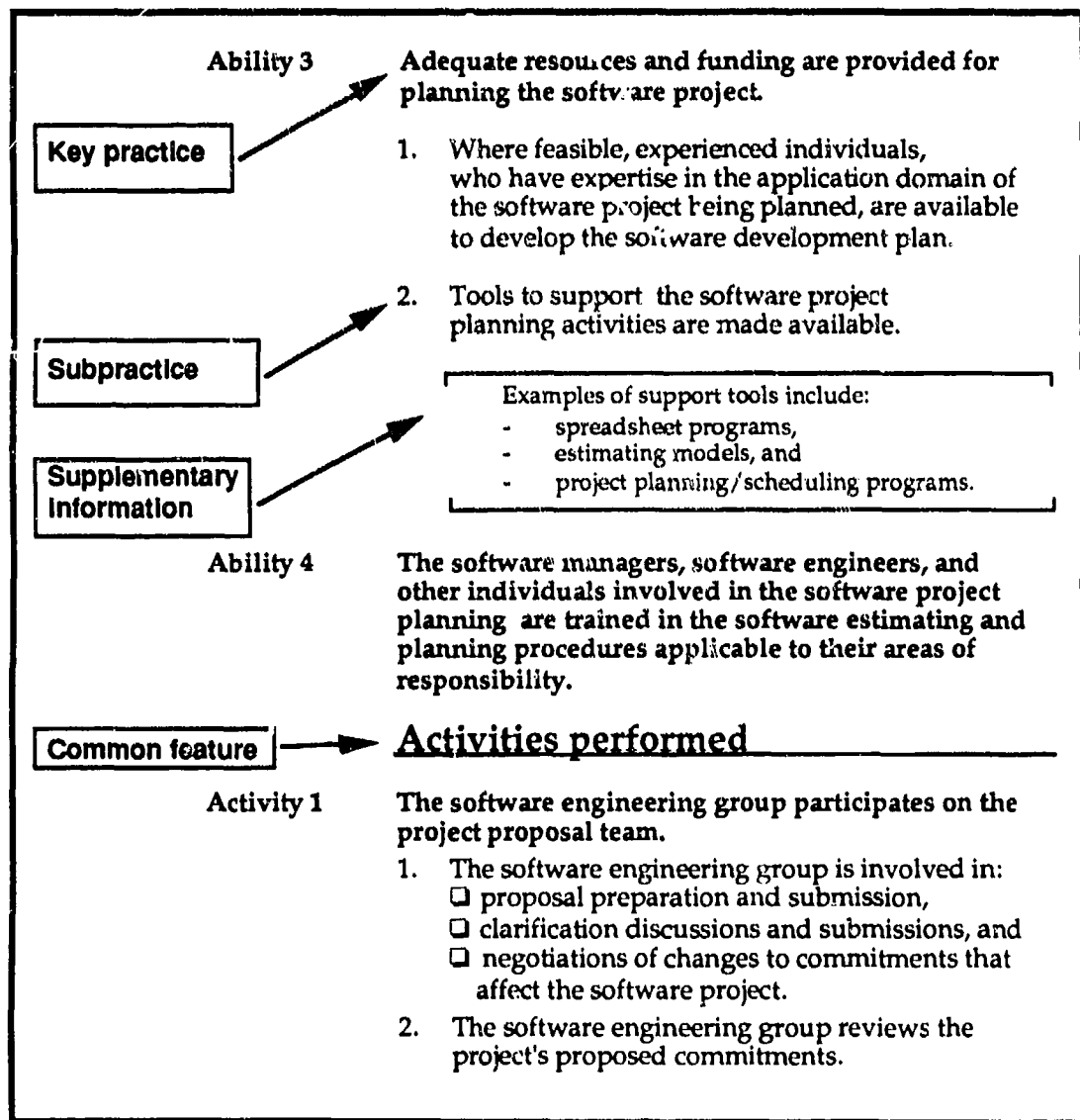


Figure 3.1 Example of Key Practice Statements

Using the Key Practice Pages

4 Interpreting the CMM

4.1 Interpreting the Key Practices

The intention in setting down the key practices is not to require or espouse a specific model of the software life cycle, a specific organizational structure, a specific separation of responsibilities, or a specific management and technical approach to development. The intention, rather, is to provide a description of the essential elements of an effective software process.

The key practices are intended to communicate principles that apply to a wide variety of projects and organizations, that are valid across a range of typical software applications, and that will remain valid over time. Therefore, the approach is to describe the principles and leave their implementation up to each organization, according to its culture and the experiences of its managers and technical staff.

Although the key practices are meant to be independent of any particular implementation, specific terms and examples are consistently used in stating the key practices to improve clarity. This section describes the conventions used in the CMM for roles, responsibilities, relationships, products, and activities. Organizations using the key practices should be aware of these conventions and map them appropriately to their own organization, project, and business environment.

The glossary in Appendix B contains definitions of terms, including those described in this section and others.

4.2 Interpreting the Common Features

Within each common feature of the key practices, certain phrases and conventions were used to provide continuity and consistency between the

Interpreting the CMM

key process areas. The major structural conventions are described below, arranged by common feature.

4.2.1 Commitment to Perform

Policy Statements Where policy statements are used, they generally refer to the project following a written, organizational policy for the practices of that key process area. This is to emphasize the connection between organizational commitment and the projects that are actually performing the work.

The subpractices for the policy statement generally summarize activities that are covered later in the key process area and are particularly suitable to institutionalization via a written policy.

In some key process areas (e.g., Organization Process Focus), the focus of the activities for the key process area is the organization, not the project. In those cases, the policy statement is reworded and refers to the organization following a written policy.

Leadership In some key process areas, Commitment to Perform contains a statement that addresses the assignment of a leadership role (e.g., project software manager) or that describes particular sponsorship activities, which are necessary for the key process area to be successfully institutionalized.

4.2.2 Ability to Perform

*Resources and
funding*

Most key process areas contain a key practice that reflects the need for adequate resources and funding for the activities covered by the key process area. These resources and funding, described by the subpractices, generally fall into three categories: access to special skills, adequate funding, and access to tools. Tools that may be of use in performing the activities of the key process area are listed as examples.

The word "funding" is used, rather than "budget," to emphasize that what is delivered and used is more pertinent to the actual process than what was promised.

“

Interpreting the CMM

Training

The CMM's context for the term training is somewhat broader than might normally be considered when using the term. Training is provided to make an individual proficient with specialized instruction and practice. This training may include informal as well as formal vehicles for transferring skills and knowledge to the individuals in the organization. Although classroom training is a common mechanism that many organizations use to build the skills of their employees, the CMM also accommodates other vehicles, such as facilitated video, computer aided instruction, or formal mentoring and apprenticeship programs. The Training Program key process area describes the specific practices related to these training vehicles.

Two templates to describe training are generally found throughout the CMM. At Level 2, the phrase "receive training" is used. At Levels 3 and above, the phrase "receive required training" is used. The intention in using these different templates is to recognize that training at Level 2 is not likely to have been institutionalized across the organization. At Levels 3 and above, the key practices of the Training Program key process area are expected to govern the organization's training activities.

In all the key process areas, potential training topics are expressed as example boxes, to recognize that different organizational situations are likely to drive different specific training needs.

Orientation

In some key process areas, key practices that describe orientation are found. The term orientation is used broadly to indicate less depth of skill¹¹ knowledge being transferred than would be expected via training. Orientation is an overview or introduction to a topic for those overseeing or interfacing with the individuals responsible for performing in the topic area.

Prerequisite Items

Some key process areas contain key practices that express a need for prerequisite items; for example, a software development plan is a prerequisite for Software Project Tracking and Oversight. In some cases, these are prerequisites that would be expected as outputs from the activities of another key process area. In other cases, they are items expected to be obtained from outside the realm of the software project (e.g., the system requirements allocated to software are a prerequisite for Requirements Management).

In keeping with the CMM philosophy of highlighting "key" practices, not all prerequisite items are listed for each key process area. Only those that have been found to be particularly critical for implementing the key process area are cited in the CMM.

4.2.3 Activities Performed

Of all the common features, Activities Performed shows the greatest amount of structural variability, because the implementation activities for the key process areas vary in level of detail, organizational focus (e.g., project or organization), and need for planning and documentation. Some generalizations are highlighted below.

Interpreting the CMM

Types of plans

Two major types of plans are described in the key practices: formal plans (e.g., software development plan, software quality assurance plan, and software configuration management plan) and informal plans (e.g., peer review plan, risk management plan, and technology management plan).

The informal plans will typically be documented as part of a formal plan (e.g., the peer review plan may be documented as part of the software development plan) or as an adjunct to a formal plan (e.g., peer review schedules may be a section in the software development plan). Formal plans require a high degree of management commitment, both from the standpoint of creating them and ensuring that they are followed. In contractual environments, these plans are usually deliverable to the customer who contracted the effort.

Formal plans

In cases where formal plans are called out, there are usually two key practices that specifically address the planning activities: a key practice that requires that the plan be developed or revised according to a documented procedure, and one that requires that the activities of the key process area be based on the plan.

The subpractices referring to a documented procedure generally cover what the inputs to the plan need to be, as well as the expected steps for obtaining commitment and support required for the plan. These subpractices identify the typical reviewers of the plan. They also highlight what levels of approval would be expected.

The subpractices that refer to the plan being the basis for activities describe the expected contents of the plan under discussion. Depending on the type of plan and need for organizational flexibility in covering the general topics of the plan, varying levels of detail are provided to describe the plan's contents.

Informal plans

Informal plans are usually described by a single key practice. The subpractices include information about the contents of the plan as well as the procedure for developing or revising the plan.

Interpreting the CMM

*According to a
documented
procedure*

A documented procedure is usually needed so that the individuals responsible for a task or activity are able to perform it in a repeatable way and so that others with general knowledge of the area will be able to learn and perform the task or activity in the same way. This is one aspect of institutionalizing a process.

The formality and level of detail of a documented procedure can vary significantly, from a hand-written individual desk procedure to a formal organizational standard operating procedure. The formality and level of detail depends on who will perform the task or activity (e.g., individual or team), how often it is performed, the importance and intended use of the results, and the intended recipients of the results.

*System
requirements
allocated to software*

The system requirements allocated to software, usually referred to as the "allocated requirements" in the CMM, are the subset of the system requirements that are to be implemented in the software components of the system. The allocated requirements are a primary input to the software development plan. Software requirements analysis elaborates and refines the allocated requirements and results in software requirements which are documented.

Customer requirements involve a complete system, not just software. In the CMM, discussion of customer requirements centers on those customer requirements to be implemented in software. The allocation of system requirements to hardware, software, etc., is typically done by a system engineering group as part of the overall system design. The system requirements allocated to the software project are usually referred to as the "allocated requirements" in the CMM and include both technical requirements (functionality, performance, etc.) and nontechnical requirements (delivery dates, cost, etc.).

Interpreting the CMM

*Customer-supplier
relationship*

The customer may be internal or external to the organization. An example of an internal customer is a marketing group; an example of an external customer is the DoD. The user may also differ from the customer, as is typically the case in the DoD contracting environment. The CMM is expressed in terms of an external customer who is procuring a system with a critical software component.

Where necessary, the boundaries between groups, as stated in the CMM, must be appropriately interpreted. For example, in software-only procurements, there may be no system engineering group between the customer and the software engineering group. In such a case, customer requirements, system requirements, and allocated requirements may be synonymous, and the responsibilities of the system engineering group will be divided between the customer and the software engineering group.

*Tracking and
taking corrective
action versus
managing*

In Software Project Tracking and Oversight at Level 2, many of the key practices use the phrase, "... is tracked... corrective actions are taken as appropriate." In Integrated Software Management at Level 3, many of the similar key practices use the phrase, "is managed." This difference in wording reflects the project's lack of a completely defined software process at Level 2. Management actions are likely to be reactions to actual problems. At Level 3, the project has a complete defined software process, and the relationships between the various software work products, tasks, and activities are well defined. Management is better able to anticipate problems and proactively prevent them from occurring. When interventions are required, the effect on the entire software process is understood, and these interventions can be more effectively defined and applied.

*Reviewed versus
undergoes peer
reviews*

At a review, a software work product, or set of work products, is presented to managers, the customer, end users, or other interested individuals for their comment or approval. Reviews typically occur at the end of a task. At a peer review, a software work product, or set of work products, is presented to the producer's colleagues to identify defects. Managers, the customer, and end users are typically not present in a peer review. Peer reviews are an integral, in-process part of a task. They are performed so that defects can be removed early, leading to higher productivity and high-quality products. Some software work products will be reviewed; some will undergo peer review; and some will undergo both peer reviews and reviews.

Interpreting the CMM

*Placed under
configuration
management versus
managed and
controlled*

Some software work products, e.g., the software design and the code, should have baselines established at predetermined points. These baselines are formally reviewed and agreed on and serve as the basis for further development. A rigorous change control process is applied to baselined items. These baselines provide control and stability when interacting with the customer. This is sometimes referred to as baseline configuration management. The phrase "placed under configuration management" is used for such software work products.

When control of the configuration is exercised by the developers, it is usually referred to as developmental configuration management. Some items under developmental configuration management may be placed under baseline configuration management at predetermined points in their development. The phrase "placed under configuration management" can be interpreted as extending to developmental configuration management, but a valid minimal interpretation is that only baseline configuration management is required.

Some software work products, such as estimates or the software development plan, which may not have to be under configuration management, still need to be "managed and controlled." This phrase is used to characterize the process of identifying and defining software work products that are not part of a baseline and, therefore, are not placed under configuration management but that must be controlled for the project to proceed in a disciplined manner. "Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

4.2.4 Measurement and Analysis

The key practices in the Measurement and Analysis common feature describe basic measurement practices that are necessary to determine status related to the Activities Performed common feature of the key practices. Measurements that are inherently part of the activities of the key process area are contained under the Activities Performed common feature.

Examples of suggested measurements are expressed as supplementary information, because the variability in project environments may lead to different measurement needs and approaches.

4.2.5 Verifying Implementation

The Verifying Implementation common feature generally contains key practices that relate to oversight by senior management and project management, as well as specific verification activities that the software quality assurance group or others are expected to perform to verify that the key practices are being performed properly.

Interpreting the CMM

Senior management oversight on a periodic basis

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

The scope and content of senior management reviews will greatly depend on which senior manager is involved in the review. Reviews by the senior manager responsible for all software activities of an organization are expected to occur on a different schedule, and address different topics, from a review by the senior executive of the entire organization. Senior management reviews would also be expected to cover different topics, or similar topics at a higher level of abstraction, than project management oversight reviews.

*Project
management
oversight on both a
periodic and event-
driven basis*

The template "both on a periodic and event-driven basis" is used in these key practices to emphasize that projects have needs for different types of review at different stages and depending on the project characteristics. Project management should maintain an ongoing awareness of the status of the software effort and be informed when significant events on the software project occur. Examples include project management participation in formal reviews, such as critical design reviews, as well as reviews which encompass process issues such as status of process improvement planning and resolution of process non-compliance issues.

At the project level, project management oversight is expected to be at a more detailed level than that of senior management, reflecting project management's more active involvement in the operational aspects of a project.

*Software quality
assurance activities*

The particular activities that are considered appropriate for review and/or audit by the software quality assurance (SQA) group are described as a key practice. There are particular cases where SQA verification activities are not described, such as for the Training Program and Intergroup Coordination key process areas. These are key process areas that are at the boundary between the software project and the organization, where the SQA group would not be expected to have authority.

4.3 Interpreting Software Process Definition

Software process definition is fundamental for achieving higher levels of maturity. This section discusses aspects of software process definition which are helpful in using the key practices related to process definition, beginning with Organization Process Definition at Level 3.

A fundamental concept of process definition in the CMM is the organization's standard software process. An organization's standard software process is the operational definition of the basic process that guides the establishment of a common software process across the software projects in the organization. It describes the fundamental software process elements that each software project is expected to incorporate into its defined software process. It also describes the relationships (e.g., ordering and interfaces) between these software process elements. It establishes a consistent way of performing the software activities across the organization and is essential for long-term stability and improvement.

At the organizational level, the organization's standard software process needs to be described, managed, controlled, and improved in a formal manner. At the project level, the emphasis is on the useability of the project's defined software process and the value it adds to the project. A project's defined software process is the operational definition of the software process used by the project. The project's defined software process is a well-characterized and understood software process, described in terms of software standards, procedures, tools, and methods. It is developed by tailoring the organization's standard software process to fit the specific characteristics of the project.

The key practices in Organization Process Definition are presented using terms that reflect an approach to process definition that supports both stability and flexibility. This approach is depicted in Figure 4.1, and its key elements are described in the following paragraphs.

4.3.1 Process Definition Concepts

A fundamental concept that supports the approach taken by the SEI in its process definition work is that processes can be developed and maintained in a way similar to the way products are developed and maintained. There must be:

- ☐ requirements that define what process is to be described,
- ☐ an architecture and design that provide information on how the process will be defined,
- ☐ implementation of the process design in a project or organizational situation,
- ☐ validation of the process description via measurement, and
- ☐ deployment of the process into widespread operation within the organization or project for which the process is intended.

Using the analogy of product development, a framework for software process development and maintenance has evolved that translates these concepts into ones which are more specific to the process development discipline (similar to the specificity of terminology used for developing real-time embedded systems versus management information systems). The key elements of this framework are illustrated in Figure 4.1 and described briefly below.

For further reading on the concepts of process definition that are being developed within the process engineering community, refer to the paper, "Software Process Development and Enactment: Concepts and Definitions" [Feiler 92].

Interpreting the CMM

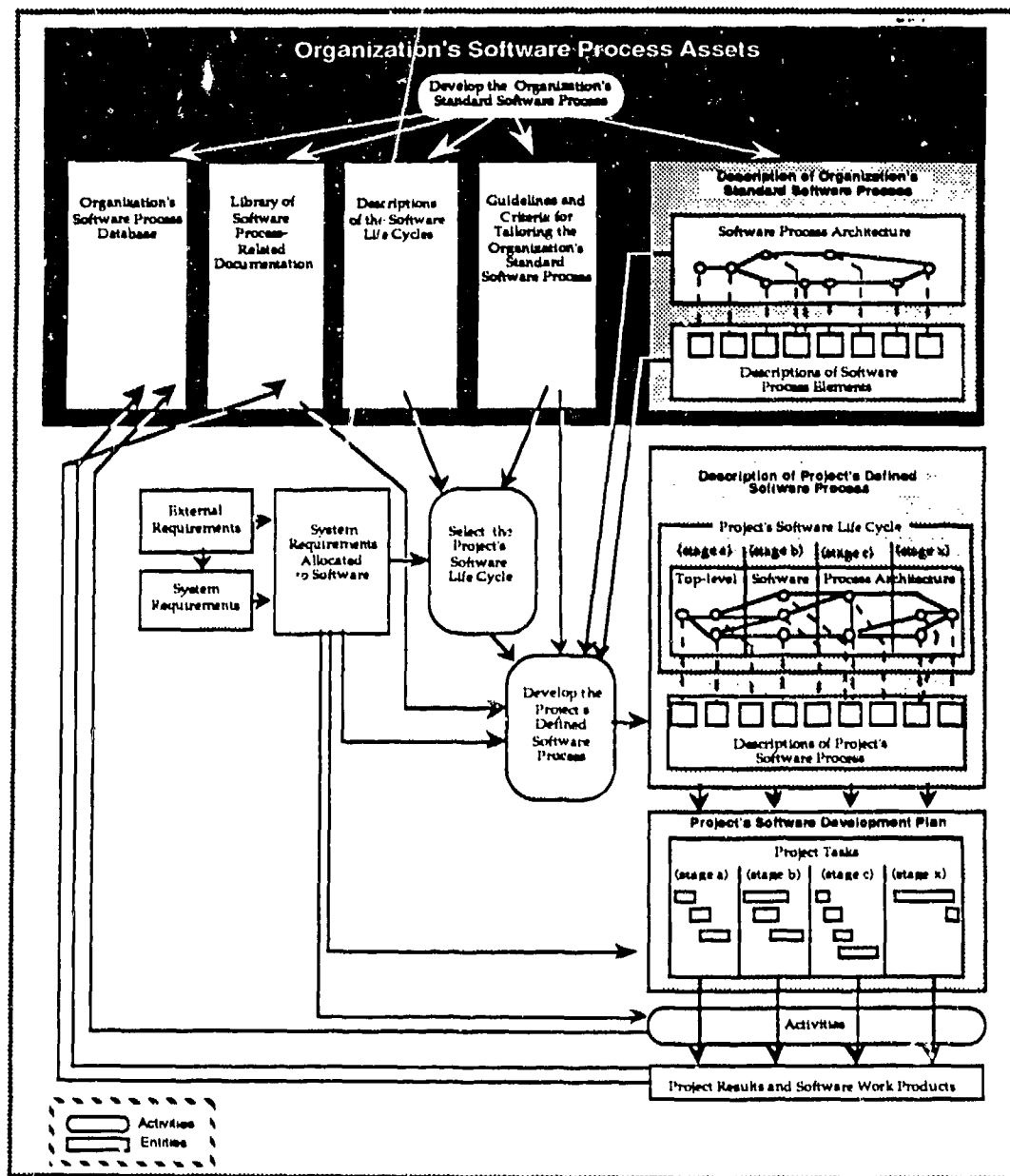


Figure 4.1 Conceptual Software Process Framework Used in the CMM

4.3.2 Concepts Related to the Organization's Software Process Assets

*Organization's
software process
assets*

The organization establishes and maintains a set of software process assets as shown in Figure 4.1. These software process assets include:

- ☐ the organization's standard software process (including the software process architecture and software process elements),
- ☐ the descriptions of software life cycles approved for use,
- ☐ the guidelines and criteria for tailoring the organization's standard software process,
- ☐ the organization's software process database, and
- ☐ the library of software process-related documentation.

The software process assets are available for use by the projects in developing, maintaining, and implementing their defined software process.

An organization may bundle the software process assets in many ways, depending on its approach to establishing its standard software process. For example, the description of the software life cycle may be an integral part of the organization's standard software process. Another example is that parts of the library of software process-related documentation may be stored in the organization's software process database.

Interpreting the CMM

*Organization's
standard software
process*

An organization's standard software process is the operational definition of the basic process that guides the establishment of a common software process across the software projects in the organization. It describes the fundamental software process elements that each software project is expected to incorporate into its defined software process. It also describes the relationships (e.g., ordering and interfaces) between these software process elements. It guides the establishment of a common software process across the software development and maintenance projects in the organization.

The relationship between software process elements is sometimes referred to as a "software process architecture."

The organization's standard software process forms the basis for the projects' defined software processes. It provides continuity in the organization's process activities and is the reference for the measurements and long-term improvement of the software processes used in the organization.

*Software process
architecture*

The software process architecture is a high-level (i.e., summary) description of the organization's standard software process. It describes the ordering, interfaces, interdependencies, and other relationships between the software process elements of the organization's standard software process. It also describes the interfaces, dependencies, and other relationships to other external processes (e.g., system engineering, hardware engineering, and contract management).

*Software process
element*

A software process element is a constituent element of a software process description. Each process element covers a well-defined, bounded, closely related set of tasks (e.g., software estimating element, software design element, coding element, and peer review element). The descriptions of the process elements may be templates to be filled in, fragments to be completed, abstractions to be refined, or complete descriptions to be modified or used unmodified.

*Description of
software life cycles
approved for use*

A software life cycle is the period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept stage, requirements stage, design stage, implementation stage, test stage, installation and checkout stage, operation and maintenance stage, and sometimes, retirement stage [IEEE-STD-610].

Because an organization may be producing software for a variety of contractual and/or commercial customers and users, one software life cycle may not be appropriate for all situations. Therefore, the organization may identify more than one software life cycle for use by the projects. These software life cycles are typically obtained from software engineering literature and may be modified for the organization. These software life cycles are available to be used, in combination with the organization's standard software process, in developing a project's defined software process.

Interpreting the CMM

Guidelines and criteria for tailoring

The organization's standard software process is described at a general level that may not be directly usable by a project. Guidelines are established to guide the software projects in (1) selecting a software life cycle from those approved for use and (2) tailoring and elaborating the organization's standard software process and the selected software life cycle to fit the specific characteristics of the project.

These guidelines and criteria help ensure that there is a common basis across all software projects for planning, implementing, measuring, analyzing, and improving the projects' defined software processes.

Organization's software process database

The organization's software process database is a database established to collect and make available data on the software processes and resulting software work products, particularly as they relate to the organization's standard software process. The database contains or references both the actual measurement data and the related information needed to understand the measurement data and assess it for reasonableness and applicability.

Examples of process and work product data include estimates of software size, effort, and cost; actual data on software size, effort, and cost; productivity data; peer review coverage and efficiency; and number and severity of defects found in the software code.

*Library of software
process-related
documentation*

A library of software process-related documentation is established to (1) store process documents that are potentially useful to other current and future projects, particularly as they relate to the organization's standard software process, and (2) make them available for sharing across the organization. This library contains example documents and document fragments, which are expected to be of use to future projects when they are tailoring the organization's standard software process. The examples may cover subjects such as a project's defined software process, standards, procedures, software development plans, measurement plans, and process training materials. This library is an important resource that can help to reduce the amount of effort required to start a new project, by providing examples of successful projects as a starting point.

4.3.3 Concepts Related to the Project's Defined Software Process

*Description of
project's defined
software process*

The description of the project's defined software process is the operational definition of the software process used by the project. The project's defined software process is a well-characterized and understood software process, described in terms of software standards, procedures, tools, and methods. It is developed by tailoring the organization's standard software process to fit the specific characteristics of the project.

This tailoring includes selecting a software life cycle from those approved by the organization and modifying the organization's standard software process to fit the specific characteristics of the project.

The project's defined software process provides the basis for planning, performing, and improving the activities of the managers and technical staff performing the project's tasks and activities. It is possible for a project to have more than one defined software process (e.g., for the operational software and for the test support software) or to have one defined software process for two or more similar projects.

Stages

A stage is a partition of the software effort that is of a manageable size and that represents a meaningful and measurable set of related tasks which are performed by the project. A stage is usually considered a subdivision of a software life cycle and is often ended with a formal review prior to the onset of the following stage.

Tasks

The work to be performed is broken down into tasks. A task is a well-defined unit of work in the software process that provides management with a visible checkpoint into the status of the project. Tasks have readiness criteria (preconditions) and completion criteria (postconditions).

Within the context of process definition, a task is a well-defined component of a defined process. All tasks can be considered activities, but not all activities are well enough defined to be considered tasks (although an activity may include a task). Because of this, use of "task" in the Level 2 key practices is avoided and the less rigorous term "activity" is used.

Activities

An activity is any step taken or function performed, both mental and physical, toward achieving some objective. Activities include all the work the managers and technical staff do to perform the tasks of the project and organization.

Interpreting the CMM

*Software work
products (project
results)*

The results of activities and tasks primarily consist of software work products. A software work product is any artifact created as part of defining, maintaining, or using a software process, including process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to a customer or end user. Work products become an input to the next step in the process or provide archival information on the software project for use in future projects.

Examples of software work products include plans, estimates, data on actual effort, corrective action documentation, and requirements documents. The subset of software work products that are deliverable to the customer or end user are referred to as software products.

Software products

The software products are the complete set, or any of the individual items of the set, of computer programs, procedures, and associated documentation and data designated for delivery to a customer or end user. [IEEE-STD-610]

All software products are also software work products. A software work product that will not be delivered to a customer or end user is not, however, a software product.

4.3.4 Relationship Between the Project's Defined Software Process and the Software Development Plan

The description of the project's defined software process will usually not be specific enough to be performed directly. Although the description typically identifies such things as roles (i.e., who performs a task) and types of software work products needed to perform a task, it does not specify the individual who will assume the roles, the specific software work products that will be created, nor the schedule for performing the tasks and activities.

The project's software development plan, either as a single document or a collection of plans collectively referred to as a software development plan, provides the bridge between the project's defined software process (what will be done and how it will be done) and the specifics of how the project will be performed (e.g., which individuals will produce which software work products according to what schedule). The combination of the project's defined software process and its software development plan makes it possible to actually perform the process.

4.3.5 Life Cycles and the CMM

The key practices are not meant to limit the choice of a software life cycle. People who have extensively used one particular software life cycle may perceive elements of that life cycle in the organization and structure of the key practices. However, there is no intent either to encourage or preclude the use of any particular software life cycle.

The term "stage" is used to refer to a defined partition of the software project's effort, but the term should not be tied to any specific software life cycle. As it is used in the key practices, "stage" can mean rigidly sequential stages or overlapping and iterative stages.

Interpreting the CMM

4.3.6 Technology and the CMM

The key practices neither require nor preclude specific software technologies, such as prototyping, object oriented design, or reusing software requirements, design, code, or other elements.

4.3.7 Documentation and the CMM

The key practices describe a number of process-related documents, each one covering specific areas of content. The key practices do not require a one-to-one relationship between the documents named in the key practices and the actual work products of an organization or project. Nor is there an intended one-to-one relationship to documents specified by the DoD or to standards such as DOD-STD-2167A or IEEE software standards. The key practices require only that the applicable contents of these documents be part of the organization's or project's written work products.

In terms of document structure, the contents of a document referred to in the key practices could be part of a larger document. For example, an organization might have a software development plan that includes the essentials of the software risk management plan.

Alternatively, the contents of a document referred to in the key practices could be distributed over a set of documents that differ from the set named in the key practices. For example, a project might develop three documents, a software development plan, a software management plan, and a project work breakdown structure, to satisfy the key practices for a software project's software risk management, software quality assurance, and software development plans.

4.3.8 Collection and Analysis of Process Data

The key practices for the collection and analysis of process data evolve across the maturity levels.

At Level 2, the data are primarily related to the size of the project's work products, effort, and schedule, and are defined, collected, and stored separately by each project. The data are shared between projects via informal mechanisms.

At Level 3, each project has a defined software process tailored from the organization's standard software process. Data related to each project's defined software process are collected and stored in the organization's software process database. The data collected and stored may be different for each project, but the data are well defined within the organization's software process database.

At Level 4, the organization defines a standard set of measurements based on the organization's standard software process. All projects collect this standard set of measurement data, as well as other project-specific data, and store them in the organization's software process database. The data are used by the projects to quantitatively understand and stabilize the process performance of the projects' defined software processes. They are also used by the organization to establish a process capability baseline for the organization's standard software process.

At Level 5, data are used to select areas for technology and process improvements, plan these improvements, and evaluate the effects of these improvements on the organization's process capability.

4.4 Organizational Structure and Roles

Although the CMM attempts to remain independent of specific organizational structures and models, it is necessary to express the practices in the CMM consistently using terminology related to organizational structure and roles, which may differ from that followed by any specific organization. The following sections describe the various concepts related to organizations, projects, and roles that are necessary for interpreting the key practices of the CMM.

4.4.1 Organizational Roles

A role is a unit of defined responsibilities that may be assumed by one or more individuals. The following descriptions of roles are frequently used in the key practices:

Manager

A manager fulfills a role that encompasses providing technical and administrative direction and control to individuals performing tasks or activities within the manager's area of responsibility. The traditional functions of a manager include planning, resourcing, organizing, directing, and controlling work within an area of responsibility.

Senior manager

A senior manager fulfills a management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects. A senior manager also provides and protects resources for long-term improvement of the software process (e.g., a software engineering process group).

Senior management, as used in the CMM, can denote any manager who satisfies the above description, up to and including the head of the whole organization. As used in the key practices, the term senior management should be interpreted in the context of the key process area and the projects and organization under consideration. The intent is to include specifically those senior managers who are needed to fulfill the leadership and oversight roles essential to achieving the goals of the key process area.

Interpreting the CMM

Project manager

A project manager fulfills the role with total business responsibility for an entire project; the project manager is the individual who directs, controls, administers, and regulates a project building a software or hardware/software system. The project manager is the individual ultimately responsible to the customer.

In a project-oriented organizational structure, most of the people working on a project would report to the project manager, although some disciplines might have a matrixed reporting relationship. In a matrixed organizational structure, it may be only the business staff who reports to the project manager. The engineering groups would then have a matrixed reporting relationship.

*Project software
manager*

A project software manager fulfills the role with total responsibility for all the software activities for a project. The project software manager is the individual the project manager deals with in terms of software commitments and who controls all the software resources for a project.

The software engineering groups on a project would report to the project software manager, although some activities such as tools development might have a matrixed reporting relationship.

In a large project, the project software manager is likely to be a second-, third-, or fourth-line manager. In a small project or department with a single project, the project software manager might be the first-line software manager or might be at a higher level.

*First-line software
manager*

A first-line software manager fulfills the role with direct management responsibility (including providing technical direction and administering the personnel and salary functions) for the staffing and activities of a single organizational unit (e.g., a department or project team) of software engineers and other related staff.

Interpreting the CMM

Software task leader A software task leader fulfills the role of leader of a technical team for a specific task, who has technical responsibility and provides technical direction to the staff working on the task.

The software task leader usually reports to the same first-line software manager as the other people who are working on the task.

*Staff, software
engineering staff,
individuals*

Several terms are used in the CMM to denote the individuals who perform the various technical roles described in various key practices of the CMM. The staff are the individuals, including task leaders, who are responsible for accomplishing an assigned function, such as software development or software configuration management, but who are not managers.

The software engineering staff are the software technical people (e.g., analysts, programmers, and engineers), including software task leaders, who perform the software development and maintenance activities for the project, but who are not managers.

The term "individuals" as used in the key practices is qualified and bounded by the context in which the term appears (e.g., "the individual involved in managing the software subcontract").

A similar breakout of roles can be identified for other engineering groups such as system engineering or system test.

In a particular project or organization, there does not need to be a one-to-one correspondence between these roles and individuals. One person could perform in multiple roles, or each role could be performed by separate individuals.

For example, on a small, software-only project, one person might have as many as six roles: the system engineering first-line manager, the project system engineering manager, the software first-line manager, the project software manager, the project manager, and the software configuration management manager.

On a slightly larger project, one person might be the system engineering first-line manager, the project system engineering manager, and the project manager while another person might be both the first-line software manager and the project software manager. These two managers might be in the same second-line organization or in different second-line organizations.

On a large project, many roles, especially those of management, would likely be filled by separate individuals.

4.4.2 Organizational Structure

The fundamental concepts of organization, project, and group must be understood to properly interpret the key practices of the Capability Maturity Model. The following paragraphs define the use of these concepts in the CMM:

Interpreting the CMM

<i>Organization</i>	An organization is a unit within a company or other entity (e.g., government agency or branch of service) within which many projects are managed as a whole. All projects within an organization share a common top-level manager and common policies.
<i>Project</i>	A project is an undertaking requiring concerted effort, which is focused on developing and/or maintaining a specific product. The product may include hardware, software, and other components. Typically a project has its own funding, cost accounting, and delivery schedule.
<i>Group</i>	A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time.

Groups commonly referred to in the CMM are described below:

*Software
engineering group*

The software engineering group is the collection of individuals (both managers and technical staff) who have responsibility for software development and maintenance activities (i.e., requirements analysis, design, code, and test) for a project.

Groups performing software-related work, such as the software quality assurance group, the software configuration management group, and the software engineering process group, are not included in the software engineering group. These groups are considered to be one of the "other software-related groups."

*Software-related
groups*

A software-related group is the collection of individuals (both managers and technical staff) representing a software engineering discipline that supports, but is not directly responsible for, software development and/or maintenance.

Examples of software engineering disciplines include software quality assurance and software configuration management.

*Software
engineering process
group*

The software engineering process group is the group of specialists who facilitate the definition, maintenance, and improvement of the software process used by the organization. In the key practices, this group is generically referred to as "the group responsible for the organization's software process activities."

Interpreting the CMM

<i>System engineering group</i>	The system engineering group is the collection of individuals (both managers and technical staff) who have responsibility for specifying the system requirements; allocating the system requirements to the hardware, software, and other components; specifying the interfaces between the hardware, software, and other components; and monitoring the design and development of these components to ensure conformance with their specifications.
<i>System test group</i>	The system test group is the collection of individuals (both managers and technical staff) who have responsibility for planning and performing the independent system testing of the software to determine whether the software product satisfies its requirements.
<i>Software quality assurance group</i>	The software quality assurance group is the collection of individuals (both managers and technical staff) who plan and implement the project's quality assurance activities to ensure the software process steps and standards are followed. Organizational issues concerning software quality assurance are discussed in Section 4.4.3.
<i>Software configuration management group</i>	The software configuration management group is the collection of individuals (both managers and technical staff) who have responsibility for planning, coordinating, and implementing the formal configuration management activities for the software project.

Training group The training group is the collection of individuals (both managers and staff) who are responsible for coordinating and arranging the training activities for an organization. This group typically prepares and conducts most of the training courses and coordinates use of other training vehicles.

4.4.3 Independence and Organizational Structure

The organization must take care that the key practices that call for independence are appropriately interpreted and followed. This is particularly true for small projects and small organizations. The key practices call for independence when technical or organizational biases may affect the quality or risks associated with the project. For example, two practices dealing with independence are:

- ❑ The SQA group has a reporting channel to senior management that is independent of the project manager, the project's software engineering group, and the other software-related groups (Commitment 1.2 in Software Quality Assurance).
- ❑ The (system and acceptance) test cases and test procedures are planned and prepared by a test group that is independent of the software developers (Activity 7.3 in Software Product Engineering).

The need for independence of the system and acceptance testing is based on technical considerations. This independence ensures that the testers are not inappropriately influenced by the design and implementation decisions made by the software developers or maintainers.

The independence of the SQA group is necessary so its members can perform their jobs without being influenced by project schedule and cost

Interpreting the CMM

pressures. Ensuring effective operational independence without the organizational independence is difficult. For example, an employee reporting to the project manager may be reluctant to stop a test activity even though serious noncompliance issues exist.

Organizations must determine the organizational structure that will support activities that require independence, such as SQA, in the context of their strategic business goals and business environment.

Independence should:

- ☐ provide the individuals performing the SQA role with the organizational freedom to be the "eyes and ears" of senior management on the project,
- ☐ protect the individuals performing the SQA role from performance appraisal by the management of the project about which they are reporting, and
- ☐ provide senior management with confidence that objective information on the process and products of the project is being reported.

Since the key practices allow interpretation of the independence criteria, professional judgment must be exercised by the organization in determining whether the goals of the key process area are achieved.

4.5 Applying Professional Judgment

To provide a complete set of valid principles that apply to a wide range of situations, some of the key practices are intentionally stated to allow for flexibility. Throughout the key practices, nonspecific phrases like "affected groups," "as appropriate," and "as necessary" are used. The use of such nonspecific terms is generally minimized in the key practices, with

examples provided in many cases, at least for the first use of the term. These phrases may have different meanings for two different organizations, for two projects in a single organization, or for one project at different points in its life cycle. Each project or organization must clarify these phrases for its specific situation.

Clarifying these phrases requires the organization to consider the overall context in which they are used. The pertinent question is whether the specific interpretation of one of these phrases meets the goals of the key process area. Professional judgment must be used to determine whether the goals have been achieved. The glossary in Appendix B may provide guidance in interpreting these and other phrases in the key practices.

Professional judgment must also be used when interpreting the key practices and how they contribute to the goals of a key process area. In general, the key process areas describe a fundamental set of behaviors that all software organizations should exhibit, regardless of their size or their products. The key practices in the CMM, however, must be interpreted in light of a project's or organization's business environment and specific circumstances. This interpretation should be based on an informed knowledge of both the CMM and the organization and its projects. The goals of the key process areas provide a means for structuring this interpretation. If an organization's implementation of a key process area satisfies the goals, but differs significantly from the key practices, the rationale for the interpretation should be documented. A documented rationale will help assessment and evaluation teams understand why certain practices are implemented the way they are.

Applying professional judgment leads to the issue of the "goodness" of the software process. The CMM does not place "goodness" requirements on the software process, although it does establish minimal criteria for a "reasonable" process in many software environments. The objective of process management is to establish processes that are used and can act as a foundation for systematic improvement based on the organization's business needs.

Interpreting the CMM

What are the criteria for a "reasonable" software process? A reasonable software process is one that is effective in building the organizational capability and satisfies most of the requirements of a defined process. Specifically, it is practiced, documented, enforced, trained, measured, and able to improve.

If an organization established a software process for estimating that consisted of rolling the dice, would that constitute a reasonable process? It could certainly be documented and consistently followed. Some might even argue that it would be as realistic as many estimating techniques. "Rolling the dice" would, however, not be judged a reasonable estimating process by most software professionals. Since it responds only to the laws of probability, it cannot be improved.

How far is it from "rolling the dice" to documenting a process to "go ask George?" This could be a very good method for estimating. As long as George is around, it could even be consistent and repeatable. It would not, however, satisfy our criteria since it cannot be trained to other individuals. It is a person-centered process that cannot be repeated without George. It does not build an ongoing organizational capability.

Using some variant of a Delphi method (a method where experts in a subject review the issues under consideration and come to consensus on the recommendations related to the issue) for estimating would usually be judged a reasonable software process. A size estimating approach based on a Delphi method satisfies the criteria for a reasonable and effective process, even though the Delphi method is a person-centered process. An organizational capability can be based on a structured technique such as a Delphi method.

In a fundamental sense, professional judgment is necessary to make such distinctions. The difficulty lies in discriminating between compliance and goodness. The goals summarize the key practices, which, in turn, describe a reasonable software process. Complying with a reasonable process,

however, does not mean that the process is efficient in achieving its purpose. There may be many factors influencing both organization and project success. For example, a successful project that builds a product that no one buys is a failure in the commercial world.

"Goodness" attributes can only be interpreted in the context of the business environment and specific circumstances of the project and the organization. Such "goodness" judgments can be made only by the organization as part of its continuous process improvement cycle. Perfection is never achieved, and continuous process improvement never ends.

Interpreting the CMM

Capability Maturity Model

Level 2: Repeatable

Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Key process areas

Requirements Management	L2-1
Software Project Planning	L2-11
Software Project Tracking and Oversight	L2-29
Software Subcontract Management	L2-43
Software Quality Assurance	L2-59
Software Configuration Management	L2-71

Requirements Management

a key process area for Level 2: Repeatable

The purpose of Requirements Management is to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project.

Requirements Management involves establishing and maintaining an agreement with the customer on the requirements for the software project. This agreement is referred to as the "system requirements allocated to the software." The "customer" may be interpreted as the system engineering group, the marketing group, another internal organization, or an external customer. The agreement covers both the technical and nontechnical (e.g., delivery dates) requirements. The agreement forms the basis for estimating, planning, performing, and tracking the software project's activities throughout the software life cycle.

The allocation of the system requirements to software, hardware, and other system components (e.g., humans) may be performed by a group external to the software engineering group (e.g., the system engineering group), and the software engineering group may have no direct control of this allocation. Within the constraints of the project, the software engineering group takes appropriate steps to ensure that the system requirements allocated to software, which they are responsible for addressing, are documented and controlled.

To achieve this control, the software engineering group reviews the initial and revised system requirements allocated to software to resolve issues

before they are incorporated into the software project. Whenever the system requirements allocated to software are changed, the affected software plans, work products, and activities are adjusted to remain consistent with the updated requirements.

Goals

- Goal 1** System requirements allocated to software are controlled to establish a baseline for software engineering and management use.
- Goal 2** Software plans, products, and activities are kept consistent with the system requirements allocated to software.

Commitment to perform

- Commitment 1** The project follows a written organizational policy for managing the system requirements allocated to software.

The system requirements allocated to the software are referred to as "allocated requirements" in these practices.

The allocated requirements are the subset of the system requirements that are to be implemented in the software components of the system. The allocated requirements are a primary input to the software development plan. Software requirements analysis elaborates and refines the allocated requirements and results in software requirements which are documented.

(Commitment 1) This policy typically specifies that:

1. The allocated requirements are documented.
2. The allocated requirements are reviewed by:
 - ☐ the software managers, and
 - ☐ other affected groups.

Examples of affected groups include:

- system test,
- software engineering (including all subgroups, such as software design),
- system engineering,
- software quality assurance,
- software configuration management, and
- documentation support.

3. The software plans, work products, and activities are changed to be consistent with changes to the allocated requirements.

Ability to perform

Ability 1

For each project, responsibility is established for analyzing the system requirements and allocating them to hardware, software, and other system components.

Analysis and allocation of the system requirements is not the responsibility of the software engineering group, but is a prerequisite for their work.

(Ability 1)

This responsibility covers:

1. Managing and documenting the system requirements and their allocation throughout the project's life.
2. Effecting changes to the system requirements and their allocation.

Ability 2

The allocated requirements are documented.

The allocated requirements include:

1. The nontechnical requirements (i.e., the agreements, conditions, and/or contractual terms) that affect and determine the activities of the software project.

Examples of agreements, conditions, and contractual terms include:

- products to be delivered,
- delivery dates, and
- milestones.

2. The technical requirements for the software.

Examples of technical requirements include:

- end user, operator, support, or integration functions;
- performance requirements;
- design constraints;
- programming language; and
- interface requirements.

3. The acceptance criteria that will be used to validate that the software products satisfy the allocated requirements.

Ability 3

Adequate resources and funding are provided for managing the allocated requirements.

1. Individuals who have experience and expertise in the application domain and in software engineering are assigned to manage the allocated requirements.
2. Tools to support the activities for managing requirements are made available.

Examples of support tools include:

- spreadsheet programs,
- tools for configuration management,
- tools for traceability, and
- tools for test management.

Ability 4

Members of the software engineering group and other software-related groups are trained to perform their requirements management activities.

Examples of training include:

- the methods, standards, and procedures used by the project, and
- the application domain.

Activities performed

Activity 1

The software engineering group reviews the allocated requirements before they are incorporated into the software project.

1. Incomplete and missing allocated requirements are identified.

(Activity 1)

2. The allocated requirements are reviewed to determine whether they are:
 - ☐ feasible and appropriate to implement in software,
 - ☐ clearly and properly stated,
 - ☐ consistent with each other, and
 - ☐ testable.
3. Any allocated requirements identified as having potential problems are reviewed with the group responsible for analyzing and allocating system requirements, and necessary changes are made.
4. Commitments resulting from the allocated requirements are negotiated with the affected groups.

Examples of affected groups include:

- software engineering (including all subgroups, such as software design),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

Refer to Activity 6 of the Software Project Planning key process area for practices covering negotiating commitments.

Activity 2

The software engineering group uses the allocated requirements as the basis for software plans, work products, and activities.

(Activity 2)

The allocated requirements:

1. Are managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of formality than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

2. Are the basis for the software development plan.
3. Are the basis for developing the software requirements.

Activity 3

Changes to the allocated requirements are reviewed and incorporated into the software project.

1. The impact to existing commitments is assessed, and changes are negotiated as appropriate.
 - ☐ Changes to commitments made to individuals and groups external to the organization are reviewed with senior management.

Refer to Activity 4 of the Software Project Planning key process area and Activity 3 of the Software Project Tracking and Oversight key process area for practices covering commitments made external to the organization.

- ☐ Changes to commitments within the organization are negotiated with the affected groups.

(Activity 3)

Refer to Activities 5, 6, 7, and 8 of the Software Project Tracking and Oversight key process area for practices covering negotiating changes to commitments.

2. Changes that need to be made to the software plans, work products, and activities resulting from changes to the allocated requirements are:
 - ☐ identified,
 - ☐ evaluated,
 - ☐ assessed for risk,
 - ☐ documented,
 - ☐ planned,
 - ☐ communicated to the affected groups and individuals, and
 - ☐ tracked to completion.

Measurement and analysis

- Measurement 1** **Measurements are made and used to determine the status of the activities for managing the allocated requirements.**

Examples of measurements include:

- status of each of the allocated requirements;
- change activity for the allocated requirements; and
- cumulative number of changes to the allocated requirements, including total number of changes proposed, open, approved, and incorporated into the system baseline.

Verifying implementation

Verification 1 **The activities for managing the allocated requirements are reviewed with senior management on a periodic basis.**

The primary purpose of periodic reviews by senior management is to provide awareness of and insight into software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2 **The activities for managing the allocated requirements are reviewed with the project manager on both a periodic and event-driven basis.**

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3 **The software quality assurance group reviews and/or audits the activities and work products for managing the allocated requirements and reports the results.**

Refer to the Software Quality Assurance key process area.

(Verification 3) At a minimum, these reviews and/or audits verify that:

1. The allocated requirements are reviewed, and problems are resolved before the software engineering group commits to them.
2. The software plans, work products, and activities are appropriately revised when the allocated requirements change.
3. Changes to commitments resulting from changes to the allocated requirements are negotiated with the affected groups.

Software Project Planning

a key process area for Level 2: Repeatable

The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project.

Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work.

The software planning begins with a statement of the work to be performed and other constraints and goals that define and bound the software project (those established by the practices of the Requirements Management key process area). The software planning process includes steps to estimate the size of the software work products and the resources needed, produce a schedule, identify and assess software risks, and negotiate commitments. Iterating through these steps may be necessary to establish the plan for the software project (i.e., the software development plan).

This plan provides the basis for performing and managing the software project's activities and addresses the commitments to the software project's customer according to the resources, constraints, and capabilities of the software project.

Goals

- Goal 1 Software estimates are documented for use in planning and tracking the software project.
- Goal 2 Software project activities and commitments are planned and documented.
- Goal 3 Affected groups and individuals agree to their commitments related to the software project.

Commitment to perform

- Commitment 1 A project software manager is designated to be responsible for negotiating commitments and developing the project's software development plan.
- Commitment 2 The project follows a written organizational policy for planning a software project.

This policy typically specifies that:

1. The system requirements allocated to software are used as the basis for planning the software project.

Refer to Activity 2 of the Requirements Management key process area.

2. The software project's commitments are negotiated between:
 - ☐ the project manager,
 - ☐ the project software manager, and

(Commitment 2) ☐ the other software managers.

3. Involvement of other engineering groups in the software activities is negotiated with these groups and is documented.

Examples of other engineering groups include:

- system engineering,
- hardware engineering, and
- system test.

4. Affected groups review the software project's:

- ☐ software size estimates,
- ☐ effort and cost estimates,
- ☐ schedules, and
- ☐ other commitments.

Examples of affected groups include:

- software engineering (including all subgroups, such as software design),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

5. Senior management reviews all software project commitments made to individuals and groups external to the organization.

6. The project's software development plan is managed and controlled.

(Commitment 2)

The term "software development plan" is used throughout these practices to refer to the overall plan for managing the software project. The use of "development" terminology is not intended to exclude software maintenance or support projects and should be appropriately interpreted in the context of the individual project.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Ability to perform

Ability 1

A documented and approved statement of work exists for the software project.

1. The statement of work covers:

- ☐ scope of the work,
- ☐ technical goals and objectives,
- ☐ identification of customers and end users,

The end users referred to in these practices are the customer designated end users or representatives of the end users.

- ☐ imposed standards,
- ☐ assigned responsibilities,

(Ability 1)

- ☐ cost and schedule constraints and goals,
- ☐ dependencies between the software project and other organizations,

Examples of other organizations include:

- the customer,
- subcontractors, and
- joint venture partners.

- ☐ resource constraints and goals, and
 - ☐ other constraints and goals for development and/or maintenance.
2. The statement of work is reviewed by:
 - ☐ the project manager,
 - ☐ the project software manager,
 - ☐ the other software managers, and
 - ☐ other affected groups.
 3. The statement of work is managed and controlled.

Ability 2

Responsibilities for developing the software development plan are assigned.

1. The project software manager, directly or by delegation, coordinates the project's software planning.
2. Responsibilities for the software work products and activities are partitioned and assigned to software managers in a traceable, accountable manner.

(Ability 2)

Examples of software work products include:

- work products for delivery to the external customer or end users, as appropriate;
- work products for use by other engineering groups; and
- major work products for internal use by the software engineering group.

Ability 3

Adequate resources and funding are provided for planning the software project.

1. Where feasible, experienced individuals, who have expertise in the application domain of the software project being planned, are available to develop the software development plan.
2. Tools to support the software project planning activities are made available.

Examples of support tools include:

- spreadsheet programs,
- estimating models, and
- project planning and scheduling programs.

Ability 4

The software managers, software engineers, and other individuals involved in the software project planning are trained in the software estimating and planning procedures applicable to their areas of responsibility.

Activities performed

Activity 1

The software engineering group participates on the project proposal team.

(Activity 1)

1. The software engineering group is involved in:
 - ☐ proposal preparation and submission,
 - ☐ clarification discussions and submissions, and
 - ☐ negotiations of changes to commitments that affect the software project.
2. The software engineering group reviews the project's proposed commitments.

Examples of project commitments include:

- the project's technical goals and objectives;
- the system and software technical solution;
- the software budget, schedule, and resources; and
- the software standards and procedures.

Activity 2

Software project planning is initiated in the early stages of, and is parallel with, the overall project planning.

Activity 3

The software engineering group participates with other affected groups in the overall project planning throughout the project's life.

1. The software engineering group reviews the project-level plans.

Activity 4

Software project commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.

Activity 5

A software life cycle with predefined stages of manageable size is identified or defined.

(Activity 5)

Examples of software life cycles include:

- waterfall,
- overlapping waterfall,
- spiral,
- serial build, and
- single prototype/overlapping waterfall.

Activity 6

The project's software development plan is developed according to a documented procedure.

This procedure typically specifies that:

1. The software development plan is based on and conforms to:
 - ☐ the customer's standards, as appropriate;
 - ☐ the project's standards;
 - ☐ the approved statement of work; and
 - ☐ the allocated requirements.
2. Plans for software-related groups and other engineering groups involved in the activities of the software engineering group are negotiated with those groups, the support efforts are budgeted, and the agreements are documented.

Examples of software-related groups include:

- software quality assurance,
- software configuration management, and
- documentation support.

(Activity 6)

Examples of other engineering groups include:

- system engineering,
- hardware engineering, and
- system test.

3. Plans for involvement of the software engineering group in the activities of other software-related groups and other engineering groups are negotiated with those groups, the support efforts are budgeted, and the agreements are documented.
4. The software development plan is reviewed by:
 - ☐ the project manager,
 - ☐ the project software manager,
 - ☐ the other software managers, and
 - ☐ other affected groups.
5. The software development plan is managed and controlled.

Activity 7

The plan for the software project is documented.

In the key practices, this plan or collection of plans is referred to as the software development plan.

Refer to Activity 1 of the Software Project Tracking and Oversight key process area for practices concerning the project's use of the software development plan.

The software development plan covers:

1. The software project's purpose, scope, goals, and objectives.
2. Selection of a software life cycle.

(Activity 7)

3. Identification of the selected procedures, methods, and standards for developing and/or maintaining the software.

Examples of software standards and procedures include:

- software development planning,
- software configuration management,
- software quality assurance,
- software design,
- problem tracking and resolution, and
- software measurement.

4. Identification of software work products to be developed.
5. Size estimates of the software work products and any changes to the software work products.
6. Estimates of the software project's effort and costs.
7. Estimated use of critical computer resources.
8. The software project's schedules, including identification of milestones and reviews.
9. Identification and assessment of the project's software risks.
10. Plans for the project's software engineering facilities and support tools.

Activity 8

Software work products that are needed to establish and maintain control of the software project are identified.

Refer to Activity 4 of the Software Configuration Management key process area.

Activity 9

Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.

This procedure typically specifies that:

1. Size estimates are made for all major software work products and activities.

Examples of software size measurements include:

- function points,
- feature points,
- lines of code,
- number of requirements, and
- number of pages.

Examples of types of work products and activities for which size estimates are made include:

- operational software and support software,
- deliverable and nondeliverable work products,
- software and nonsoftware work products (e.g., documents), and
- activities for developing, verifying, and validating work products.

2. Software work products are decomposed to the granularity needed to meet the estimating objectives.
3. Historical data are used where available.
4. Size estimating assumptions are documented.
5. Size estimates are documented, reviewed, and agreed to.

(Activity 9)

Examples of groups and individuals who review and agree to size estimates include:

- the project manager,
- the project software manager, and
- the other software managers.

Activity 10

Estimates for the software project's effort and costs are derived according to a documented procedure.

This procedure typically specifies that:

1. Estimates for the software project's effort and costs are related to the size estimates of the software work products (or the size of the changes).
2. Productivity data (historical and/or current) are used for the estimates when available; sources and rationale for these data are documented.
 - ☐ The productivity and cost data are from the organization's projects when possible.
 - ☐ The productivity and cost data take into account the effort and significant costs that go into making the software work products.

Examples of significant costs that go into making the software work products include:

- direct labor expenses,
- overhead expenses,
- travel expenses, and
- computer use costs.

3. Effort, staffing, and cost estimates are based on past experience.
 - ☐ Similar projects should be used when possible.
 - ☐ Time phasing of activities is derived.

(Activity 10)

- ☐ Distributions of the effort, staffing, and cost estimates over the software life cycle are prepared.
- 4. Estimates and the assumptions made in deriving the estimates are documented, reviewed, and agreed to.

Activity 11

Estimates for the project's critical computer resources are derived according to a documented procedure.

Critical computer resources may be in the host environment, in the integration and testing environment, in the target environment, or in any combination of these.

This procedure typically specifies that:

1. Critical computer resources for the project are identified.

Examples of critical computer resources include:

- computer memory capacity,
- computer processor use, and
- communications channel capacity.

2. Estimates for the critical computer resources are related to the estimates of:
 - ☐ the size of the software work products,
 - ☐ the operational processing load, and
 - ☐ the communications traffic
3. Estimates of the critical computer resources are documented, reviewed, and agreed to.

Activity 12

The project's software schedule is derived according to a documented procedure.

(Activity 12)

This procedure typically specifies that:

1. The software schedule is related to:
 - ☐ the size estimate of the software work products (or the size of changes), and
 - ☐ the software effort and costs.
2. The software schedule is based on past experience.
 - ☐ Similar projects are used when possible.
3. The software schedule accommodates the imposed milestone dates, critical dependency dates, and other constraints.
4. The software schedule activities are of appropriate duration and the milestones are of appropriate time separation to support accuracy in progress measurement.
5. Assumptions made in deriving the schedule are documented.
6. The software schedule is documented, reviewed, and agreed to.

Activity 13

The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented.

1. The risks are analyzed and prioritized based on their potential impact to the project.
2. Contingencies for the risks are identified.

Examples of contingencies include:

- schedule buffers,
- alternate staffing plans, and
- alternate plans for additional computing equipment.

Activity 14

Plans for the project's software engineering facilities and support tools are prepared.

1. Estimates of capacity requirements for these facilities and support tools are based on the size estimates of the software work products and other characteristics.

Examples of software development facilities and support tools include:

- software development computers and peripherals,
- software test computers and peripherals,
- target computer environment software, and
- other support software.

2. Responsibilities are assigned and commitments are negotiated to procure or develop these facilities and support tools.
3. The plans are reviewed by all affected groups.

Activity 15

Software planning data are recorded.

1. Information recorded includes the estimates and the associated information needed to reconstruct the estimates and assess their reasonableness.
2. The software planning data are managed and controlled.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the software planning activities.

(Measurement 1)

Examples of measurements include:

- completions of milestones for the software project planning activities compared to the plan; and
- work completed, effort expended, and funds expended in the software project planning activities compared to the plan.

Verifying implementation

Verification 1

The activities for software project planning are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

1. The technical, cost, staffing, and schedule performance is reviewed.
2. Conflicts and issues not resolvable at lower levels are addressed.
3. Software project risks are addressed.
4. Action items are assigned, reviewed, and tracked to closure.
5. A summary report from each meeting is prepared and distributed to the affected groups and individuals.

Verification 2

The activities for software project planning are reviewed with the project manager on both a periodic and event-driven basis.

- (Verification 2)**
1. Affected groups are represented.
 2. Status and current results of the software project planning activities are reviewed against the software project's statement of work and allocated requirements.
 3. Dependencies between groups are addressed.
 4. Conflicts and issues not resolvable at lower levels are addressed.
 5. Software project risks are reviewed.
 6. Action items are assigned, reviewed, and tracked to closure.
 7. A summary report from each meeting is prepared and distributed to the affected groups and individuals.

Verification 3 **The software quality assurance group reviews and/or audits the activities and work products for software project planning and reports the results.**

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. The activities for software estimating and planning.
2. The activities for reviewing and making project commitments.
3. The activities for preparing the software development plan.
4. The standards used for preparing the software development plan.
5. The content of the software development plan.

Software Project Tracking and Oversight

a key process area for Level 2: Repeatable

The purpose of Software Project Tracking and Oversight is to provide adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.

Software Project Tracking and Oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these plans based on the actual accomplishments and results.

A documented plan for the software project (i.e., the software development plan, as described in the Software Project Planning key process area) is used as the basis for tracking the software activities, communicating status, and revising plans. Software activities are monitored by the management. Progress is primarily determined by comparing the actual software size, effort, cost, and schedule to the plan when selected software work products are completed and at selected milestones. When it is determined that the software project's plans are not being met, corrective actions are taken. These actions may include revising the software development plan to reflect the actual accomplishments and replanning the remaining work or taking actions to improve the performance.

Goals

- | | |
|---------------|---|
| Goal 1 | Actual results and performances are tracked against the software plans. |
| Goal 2 | Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans. |
| Goal 3 | Changes to software commitments are agreed to by the affected groups and individuals. |

Commitment to perform

- | | |
|---------------------|--|
| Commitment 1 | A project software manager is designated to be responsible for the project's software activities and results. |
| Commitment 2 | The project follows a written organizational policy for managing the software project. |

This policy typically specifies that:

1. A documented software development plan is used and maintained as the basis for tracking the software project.
2. The project manager is kept informed of the software project's status and issues.
3. Corrective actions are taken when the software plan is not being achieved, either by adjusting performance or by adjusting the plans.
4. Changes to the software commitments are made with the involvement and agreement of the affected groups.

(Commitment 2)

Examples of affected groups include:

- software engineering (including all subgroups, such as software design),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

5. Senior management reviews all commitment changes and new software project commitments made to individuals and groups external to the organization.

Ability to perform

Ability 1

A software development plan for the software project is documented and approved.

Refer to Activities 6 and 7 of the Software Project Planning key process area for practices covering the software development plan.

Ability 2

The project software manager explicitly assigns responsibility for software work products and activities.

The assigned responsibilities cover:

1. The software work products to be developed or services to be provided.

- (Ability 2)
2. The effort and cost for these software activities.
 3. The schedule for these software activities.
 4. The budget for these software activities.

Ability 3 Adequate resources and funding are provided for tracking the software project.

1. The software managers and the software task leaders are assigned specific responsibilities for tracking the software project.
2. Tools to support software tracking are made available.

Examples of support tools include:

- spreadsheet programs, and
- project planning/scheduling programs.

Ability 4 The software managers are trained in managing the technical and personnel aspects of the software project.

Examples of training include:

- managing technical projects;
- tracking and oversight of software size, effort, cost, and schedule; and
- managing people.

Ability 5 First-line software managers receive orientation in the technical aspects of the software project.

(Ability 5)

Examples of orientation include:

- the project's software engineering standards and procedures, and
- the project's application domain.

Activities performed

Activity 1

A documented software development plan is used for tracking the software activities and communicating status.

Refer to Activity 7 of the Software Project Planning key process area for practices covering the content of the software development plan.

This software development plan is:

1. Updated as the work progresses to reflect accomplishments, particularly when milestones are completed.
2. Readily available to:
 - ☐ the software engineering group (including all subgroups, such as software design),
 - ☐ the software managers,
 - ☐ the project manager,
 - ☐ senior management, and
 - ☐ other affected groups.

Activity 2

The project's software development plan is revised according to a documented procedure.

(Activity 2)

Refer to Activity 6 of the Software Project Planning key process area for practices covering the activities for producing the software development plan.

This procedure typically specifies that:

1. The software development plan is revised, as appropriate, to incorporate plan refinements and incorporate plan changes, particularly when plans change significantly.

Interdependencies between the system requirements allocated to software, design constraints, resources, costs, and schedule need to be reflected in all changes to the plan.

2. The software development plan is updated to incorporate all new software project commitments and changes to commitments.
3. The software development plan is reviewed at each revision.
4. The software development plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 3 Software project commitments and changes to commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.

Activity 4 Approved changes to commitments that affect the software project are communicated to the members of the software engineering group and other software-related groups.

Examples of other software-related groups include:

- software quality assurance,
- software configuration management, and
- documentation support.

Activity 5 The size of the software work products (or size of the changes to the software work products) are tracked, and corrective actions are taken as necessary.

Refer to Activity 9 of the Software Project Planning key process area for practices covering derivation of size estimates.

1. Sizes for all major software work products (or the size of the changes) are tracked.
2. Actual size of code (generated, fully tested, and delivered) is compared to the estimates documented in the software development plan.
3. Actual units of delivered documentation are compared to the estimates documented in the software development plan.
4. Overall projected size of the software work products (estimates combined with actuals) is refined, monitored, and adjusted on a regular basis.

(Activity 5)

5. Changes in size estimates of the software work products that affect software commitments are negotiated with the affected groups and are documented.

Activity 6

The project's software effort and costs are tracked, and corrective actions are taken as necessary.

Refer to Activity 10 of the Software Project Planning key process area for practices covering the derivation of cost estimates.

1. Actual expenditures of effort and costs over time and against work completed are compared to the estimates documented in the software development plan to identify potential overruns and underruns.
2. Software costs are tracked and compared to the estimates documented in the software development plan.
3. Effort and staffing are compared to the estimates documented in the software development plan.
4. Changes in staffing and other software costs that affect software commitments are negotiated with the affected groups and are documented.

Activity 7

The project's critical computer resources are tracked, and corrective actions are taken as necessary.

Refer to Activity 11 of the Software Project Planning key process area for practices covering the derivation of computer resource estimates.

1. The actual and projected use of the project's critical computer resources are tracked and compared to the estimates for each major software component as documented in the software development plan.

- (Activity 7) 2. Changes in estimates of critical computer resources that affect software commitments are negotiated with the affected groups and are documented.

Activity 8 The project's software schedule is tracked, and corrective actions are taken as necessary.

Refer to Activity 12 of the Software Project Planning key process area for practices covering derivation of the schedule.

1. Actual completion of software activities, milestones, and other commitments is compared against the software development plan.
2. Effects of late and early completion of software activities, milestones, and other commitments are evaluated for impacts on future activities and milestones.
3. Software schedule revisions that affect software commitments are negotiated with the affected groups and are documented.

Activity 9 Software engineering technical activities are tracked, and corrective actions are taken as necessary.

1. Members of the software engineering group report their technical status to their first-line manager on a regular basis.
2. Software release contents for successive builds are compared to the plans documented in the software development plan.
3. Problems identified in any of the software work products are reported and documented.
4. Problem reports are tracked to closure.

Activity 10 The software risks associated with cost, resource, schedule, and technical aspects of the project are tracked.

(Activity 10)

Refer to Activity 13 of the Software Project Planning key process area for practices covering identification of risks.

1. The priorities of the risks and the contingencies for the risks are adjusted as additional information becomes available.
2. High-risk areas are reviewed with the project manager on a regular basis.

Activity 11

Actual measurement data and replanning data for the software project are recorded.

Refer to Activity 15 of the Software Project Planning key process area for practices covering recording of project data.

1. Information recorded includes the estimates and associated information needed to reconstruct the estimates and verify their reasonableness.
2. The software replanning data are managed and controlled.
3. The software planning data, replanning data, and the actual measurement data are archived for use by ongoing and future projects.

Activity 12

The software engineering group conducts periodic internal reviews to track technical progress, plans, performance, and issues against the software development plan.

These reviews are conducted between:

1. The first-line software managers and their software task leaders.
2. The project software manager, first-line software managers, and other software managers, as appropriate.

Activity 13

Formal reviews to address the accomplishments and results of the software project are conducted at selected project milestones according to a documented procedure.

These reviews:

1. Are planned to occur at meaningful points in the software project's schedule, such as the beginning or completion of selected stages.
2. Are conducted with the customer, end user, and affected groups within the organization, as appropriate.

The end users referred to in these practices are the customer designated end users or representatives of the end users.

3. Use materials that are reviewed and approved by the responsible software managers.
4. Address the commitments, plans, and status of the software activities.
5. Result in the identification and documentation of significant issues, action items, and decisions.
6. Address the software project risks.
7. Result in the refinement of the software development plan, as necessary.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the software tracking and oversight activities.

(Measurement 1)

Examples of measurements include:

- effort and other resources expended in performing the tracking and oversight activities; and
- change activity for the software development plan, which includes changes to size estimates of the software work products, software cost estimates, critical computer resource estimates, and schedule.

Verifying implementation

Verification 1

The activities for software project tracking and oversight are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

1. The technical, cost, staffing, and schedule performance are reviewed.
2. Conflicts and issues not resolvable at lower levels are addressed.
3. Software project risks are addressed.
4. Action items are assigned, reviewed, and tracked to closure.
5. A summary status report from each meeting is prepared and distributed to the affected groups.

Verification 2 The activities for software project tracking and oversight are reviewed with the project manager on both a periodic and event-driven basis.

1. Affected groups are represented.
2. The technical, cost, staffing, and schedule performance is reviewed against the software development plan.
3. Use of critical computer resources is reviewed; current estimates and actual use of these critical computer resources are reported against the original estimates.
4. Dependencies between groups are addressed.
5. Conflicts and issues not resolvable at lower levels are addressed.
6. Software project risks are addressed.
7. Action items are assigned, reviewed, and tracked to closure.
8. A summary report from each meeting is prepared and distributed to the affected groups.

Verification 3 The software quality assurance group reviews and/or audits the activities and work products for software project tracking and oversight and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. The activities for reviewing and revising commitments.
2. The activities for revising the software development plan.

- (Verification 3)
3. The content of the revised software development plan.
 4. The activities for tracking the software project's cost, schedule, risks, technical and design constraints, and functionality and performance.
 5. The activities for conducting the planned technical and management reviews.

Software Subcontract Management

a key process area for Level 2: Repeatable

The purpose of Software Subcontract Management is to select qualified software subcontractors and manage them effectively.

Software Subcontract Management involves selecting a software subcontractor, establishing commitments with the subcontractor, and tracking and reviewing the subcontractor's performance and results. These practices cover the management of a software (only) subcontract, as well as the management of the software component of a subcontract that includes software, hardware, and possibly other system components.

The subcontractor is selected based on its ability to perform the work. Many factors contribute to the decision to subcontract a portion of the prime contractor's work. Subcontractors may be selected based on strategic business alliances, as well as technical considerations. The practices of this key process area address the traditional acquisition process associated with subcontracting a defined portion of the work to another organization.

When subcontracting, a documented agreement covering the technical and nontechnical (e.g., delivery dates) requirements is established and is used as the basis for managing the subcontract. The work to be done by the subcontractor and the plans for the work are documented. The standards

that are to be followed by the subcontractor are compatible with the prime contractor's standards.

The software planning, tracking, and oversight activities for the subcontracted work are performed by the subcontractor. The prime contractor ensures that these planning, tracking, and oversight activities are performed appropriately and that the software products delivered by the subcontractor satisfy their acceptance criteria. The prime contractor works with the subcontractor to manage their product and process interfaces.

Goals

- | | |
|---------------|---|
| Goal 1 | The prime contractor selects qualified software subcontractors. |
| Goal 2 | The prime contractor and the software subcontractor agree to their commitments to each other. |
| Goal 3 | The prime contractor and the software subcontractor maintain ongoing communications. |
| Goal 4 | The prime contractor tracks the software subcontractor's actual results and performance against its commitments. |

Commitment to perform

- | | |
|---------------------|---|
| Commitment 1 | The project follows a written organizational policy for managing the software subcontract. |
|---------------------|---|

(Commitment 1) This policy typically specifies that:

1. Documented standards and procedures are used in selecting software subcontractors and managing the software subcontracts.
2. The contractual agreements form the basis for managing the subcontract.
3. Changes to the subcontract are made with the involvement and agreement of both the prime contractor and the subcontractor.

Commitment 2 A subcontract manager is designated to be responsible for establishing and managing the software subcontract.

1. The subcontract manager is knowledgeable and experienced in software engineering or has individuals assigned who have that knowledge and experience.
2. The subcontract manager is responsible for coordinating the technical scope of work to be subcontracted and the terms and conditions of the subcontract with the affected parties.

The project's system engineering group and software engineering group define the technical scope of the work to be subcontracted.

The appropriate business function groups, such as purchasing, finance, and legal, establish and monitor the terms and conditions of the subcontract.

3. The subcontract manager is responsible for:
 - ☐ selecting the software subcontractor,
 - ☐ managing the software subcontract, and
 - ☐ arranging for the post-subcontract support of the subcontracted products.

Ability to perform

Ability 1

Adequate resources and funding are provided for selecting the software subcontractor and managing the subcontract.

1. Software managers and other individuals are assigned specific responsibilities for managing the subcontract.
2. Tools to support managing the subcontract are made available.

Examples of support tools include:

- estimating models,
- spreadsheet programs, and
- project management and scheduling programs.

Ability 2

Software managers and other individuals who are involved in establishing and managing the software subcontract are trained to perform these activities.

Examples of training include:

- preparing and planning for software subcontracting,
- evaluating a subcontract bidder's software process capability,
- evaluating a subcontract bidder's software estimates and plans,
- selecting a subcontractor, and
- managing a subcontract.

Ability 3

Software managers and other individuals who are involved in managing the software subcontract receive orientation in the technical aspects of the subcontract.

(Ability 3)

Examples of orientation include:

- application domain,
- software technologies being applied,
- software tools being used,
- methodologies being used,
- standards being used, and
- procedures being used.

Activities performed

Activity 1

The work to be subcontracted is defined and planned according to a documented procedure.

This procedure typically specifies that:

1. The software products and activities to be subcontracted are selected based on a balanced assessment of both technical and nontechnical characteristics of the project.
 - ☐ The functions or subsystems to be subcontracted are selected to match the skills and capabilities of potential subcontractors.
 - ☐ The specification of the software products and activities to be subcontracted is determined based on a systematic analysis and appropriate partitioning of the system and software requirements.
2. The specification of the work to be subcontracted and the standards and procedures to be followed are derived from the project's:
 - ☐ statement of work,
 - ☐ system requirements allocated to software,
 - ☐ software requirements,
 - ☐ software development plan, and
 - ☐ software standards and procedures.
3. A subcontract statement of work is:

(Activity 1)

- ☐ prepared,
- ☐ reviewed,
- ☐ agreed to,

Examples of individuals who review and agree to the subcontract statement of work include:

- the project manager,
- the project software manager,
- the responsible software managers,
- the software configuration management manager,
- the software quality assurance manager, and
- the subcontract manager.

- ☐ revised when necessary, and
- ☐ managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Refer to Ability 1 of the Software Project Planning key process area for practices covering typical contents of the statement of work.

4. A plan for selecting a subcontractor is prepared concurrent with the subcontract statement of work and is reviewed, as appropriate.

Activity 2

The software subcontractor is selected, based on an evaluation of the subcontract bidders' ability to perform the work, according to a documented procedure.

This procedure covers the evaluation of:

1. Proposals submitted for the planned subcontract.
2. Prior performance records on similar work, if available.
3. The geographic locations of the subcontract bidders' organizations relative to the prime contractor.

Effective management of some subcontracts may require frequent face-to-face interactions.

4. Software engineering and software management capabilities.

An example of a method to evaluate subcontractors' capabilities is the SEI Software Capability Evaluation method.

5. Staff available to perform the work.
6. Prior experience in similar applications, including software expertise on the subcontractor's software management team.
7. Available resources.

Examples of resources include:

- facilities,
 - hardware,
 - software, and
 - training.

Activity 3

The contractual agreement between the prime contractor and the software subcontractor is used as the basis for managing the subcontract.

The contractual agreement documents:

1. The terms and conditions.
2. The statement of work.

Refer to Ability 1 of the Software Project Planning key process area for practices covering the typical contents of a statement of work.

3. The requirements for the products to be developed.
4. The list of dependencies between the subcontractor and the prime contractor.
5. The subcontracted products to be delivered to the prime contractor.

Examples of products include:

- source code,
- software development plan,
- simulation environment,
- design documentation, and
- acceptance test plan.

6. The conditions under which revisions to products are to be submitted.
7. The acceptance procedures and acceptance criteria to be used in evaluating the subcontracted products before they are accepted by the prime contractor.

(Activity 3)

8. The procedures and evaluation criteria to be used by the prime contractor to monitor and evaluate the subcontractor's performance.

Activity 4

A documented subcontractor's software development plan is reviewed and approved by the prime contractor.

1. This software development plan covers (directly or by reference) the appropriate items from the prime contractor's software development plan.

In some cases, the prime contractor's software development plan, may include the software development plan for the subcontractor, and no separate subcontractor's software development plan is needed.

Refer to Activity 7 of the Software Project Planning key process area for practices covering content of the project's software development plan.

Activity 5

A documented and approved subcontractor's software development plan is used for tracking the software activities and communicating status.

Activity 6

Changes to the software subcontractor's statement of work, subcontract terms and conditions, and other commitments are resolved according to a documented procedure.

1. This procedure typically specifies that all affected groups of both the prime contractor and the subcontractor are involved.

Activity 7

The prime contractor's management conducts periodic status/coordination reviews with the software subcontractor's management.

(Activity 7)

1. The subcontractor is provided with visibility of the needs and desires of the product's customers and end users, as appropriate.

The end users referred to in these practices are the customer designated end users or representatives of the end users.

2. The subcontractor's technical, cost, staffing, and schedule performance is reviewed against the subcontractor's software development plan.
3. Computer resources designated as critical for the project are reviewed; the subcontractor's contribution to the current estimates are tracked and compared to the estimates for each software component as documented in the subcontractor's software development plan.
4. Critical dependencies and commitments between the subcontractor's software engineering group and other subcontractor groups are addressed.
5. Critical dependencies and commitments between the prime contractor and the subcontractor are addressed.
 - ☐ Subcontractor commitments to the prime contractor and prime contractor commitments to the subcontractor are both reviewed.
6. Nonconformance to the subcontract is addressed.
7. Project risks involving the subcontractor's work are addressed.
8. Conflicts and issues not resolvable internally by the subcontractor are addressed.
9. Action items are assigned, reviewed, and tracked to closure.

Activity 8

Periodic technical reviews and interchanges are held with the software subcontractor.

(Activity 8)

These reviews:

1. Provide the subcontractor with visibility of the customer's and end users' needs and desires, as appropriate.
2. Monitor the subcontractor's technical activities.
3. Verify that the subcontractor's interpretation and implementation of the technical requirements conform to the prime contractor's requirements.
4. Verify that commitments are being met.
5. Verify that technical issues are resolved in a timely manner.

Activity 9

Formal reviews to address the subcontractor's software engineering accomplishments and results are conducted at selected milestones according to a documented procedure.

This procedure typically specifies that:

1. Reviews are preplanned and documented in the statement of work.
2. Reviews address the subcontractor's commitments for, plans for, and status of the software activities.
3. Significant issues, action items, and decisions are identified and documented.
4. Software risks are addressed.
5. The subcontractor's software development plan is refined, as appropriate.

Activity 10

The prime contractor's software quality assurance group monitors the subcontractor's software quality assurance activities according to a documented procedure.

(Activity 10)

This procedure typically specifies that:

1. The subcontractor's plans, resources, procedures, and standards for software quality assurance are periodically reviewed to ensure they are adequate to monitor the subcontractor's performance.
2. Regular reviews of the subcontractor are conducted to ensure the approved procedures and standards are being followed.
 - ☐ The prime contractor's software quality assurance group spot checks the subcontractor's software engineering activities and products.
 - ☐ The prime contractor's software quality assurance group audits the subcontractor's software quality assurance records, as appropriate
3. The subcontractor's records of its software quality assurance activities are periodically audited to assess how well the software quality assurance plans, standards, and procedures are being followed.

Activity 11

The prime contractor's software configuration management group monitors the subcontractor's activities for software configuration management according to a documented procedure.

This procedure typically specifies that:

1. The subcontractor's plans, resources, procedures, and standards for software configuration management are reviewed to ensure they are adequate.
2. The prime contractor and the subcontractor coordinate their activities on matters relating to software configuration management to ensure that the subcontractor's products can be readily integrated or incorporated into the project environment of the prime contractor.
3. The subcontractor's software baseline library is periodically audited to assess how well the standards and procedures for software configuration management are being followed and how effective they are in managing the software baseline.

Activity 12

The prime contractor conducts acceptance testing as part of the delivery of the subcontractor's software products according to a documented procedure.

This procedure typically specifies that:

1. The acceptance procedures and acceptance criteria for each product are defined, reviewed, and approved by both the prime contractor and the subcontractor prior to the test.
2. The results of the acceptance tests are documented.
3. An action plan is established for any software product that does not pass its acceptance test.

Activity 13

The software subcontractor's performance is evaluated on a periodic basis, and the evaluation is reviewed with the subcontractor.

Evaluation of the subcontractor's performance provides an opportunity for the subcontractor to obtain feedback on whether or not it is satisfying its customer's (i.e., the prime contractor's) needs. A mechanism such as performance award fee reviews provides this type of feedback, as opposed to the periodic coordination and technical reviews which occur throughout the project. Documentation of these evaluations also acts as input for future subcontractor selection activities.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the activities for managing the software subcontract.

(Measurement 1)

Examples of measurements include:

- costs of the activities for managing the subcontract compared to the plan,
- actual delivery dates for subcontracted products compared to the plan, and
- actual dates of prime contractor deliveries to the subcontractor compared to the plan.

Verifying implementation

Verification 1

The activities for managing the software subcontract are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of and insight into software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2

The activities for managing the software subcontract are reviewed with the project manager on both a periodic and event-driven basis.

(Verification 2)

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3

The software quality assurance group reviews and/or audits the activities and work products for managing the software subcontract and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. The activities for selecting the subcontractor.
2. The activities for managing the software subcontract.
3. The activities for coordinating configuration management activities of the prime contractor and subcontractor.
4. The conduct of planned reviews with the subcontractor.
5. The conduct of reviews that establish completion of key project milestones or stages for the subcontract.
6. The acceptance process for the subcontractor's software products.

Software Quality Assurance

a key process area for Level 2: Repeatable

The purpose of Software Quality Assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built.

Software Quality Assurance involves reviewing and auditing the software products and activities to verify that they comply with the applicable procedures and standards and providing the software project and other appropriate managers with the results of these reviews and audits.

The software quality assurance group works with the software project during its early stages to establish plans, standards, and procedures that will add value to the software project and satisfy the constraints of the project and the organization's policies. By participating in establishing the plans, standards, and procedures, the software quality assurance group helps ensure they fit the project's needs and verifies that they will be usable for performing reviews and audits throughout the software life cycle. The software quality assurance group reviews project activities and audits software work products throughout the life cycle and provides management with visibility as to whether the software project is adhering to its established plans, standards, and procedures.

Compliance issues are first addressed within the software project and resolved there if possible. For issues not resolvable within the software

project, the software quality assurance group escalates the issue to an appropriate level of management for resolution.

This key process area covers the practices for the group performing the software quality assurance function. The practices identifying the specific activities and work products that the software quality assurance group reviews and/or audits are generally contained in the Verifying Implementation common feature of the other key process areas.

Goals

- | | |
|---------------|---|
| Goal 1 | Software quality assurance activities are planned. |
| Goal 2 | Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively. |
| Goal 3 | Affected groups and individuals are informed of software quality assurance activities and results. |
| Goal 4 | Noncompliance issues that cannot be resolved within the software project are addressed by senior management. |

Commitment to perform

- | | |
|---------------------|---|
| Commitment 1 | The project follows a written organizational policy for implementing software quality assurance (SQA). |
|---------------------|---|

This policy typically specifies that:

1. The SQA function is in place on all software projects.

(Commitment 1) 2. The SQA group has a reporting channel to senior management that is independent of:

- ☐ the project manager,
- ☐ the project's software engineering group, and
- ☐ the other software-related groups.

Examples of other software-related groups include:

- software configuration management, and
- documentation support.

Organizations must determine the organizational structure that will support activities that require independence, such as SQA, in the context of their strategic business goals and business environment.

Independence should:

- provide the individuals performing the SQA role with the organizational freedom to be the "eyes and ears" of senior management on the software project;
- protect the individuals performing the SQA role from performance appraisal by the management of the software project they are reviewing; and
- provide senior management with confidence that objective information on the process and products of the software project is being reported.

3. Senior management periodically reviews the SQA activities and results.

Ability to perform

Ability 1

A group that is responsible for coordinating and implementing SQA for the project (i.e., the SQA group) exists.

(Ability 1)

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time. Considerations when implementing a group include assigned tasks or activities, the size of the project, the organizational structure, and the organizational culture. Some groups, such as the software quality assurance group, are focused on project activities, and others, such as the software engineering process group, are focused on organization-wide activities.

Ability 2

Adequate resources and funding are provided for performing the SQA activities.

1. A manager is assigned specific responsibilities for the project's SQA activities.
2. A senior manager, who is knowledgeable in the SQA role and has the authority to take appropriate oversight actions, is designated to receive and act on software noncompliance items.
 - ☐ All managers in the SQA reporting chain to the senior manager are knowledgeable in the SQA role, responsibilities, and authority.
3. Tools to support the SQA activities are made available.

Examples of support tools include:

- workstations,
- database programs,
- spreadsheet programs, and
- auditing tools.

Ability 3

Members of the SQA group are trained to perform their SQA activities.

(Ability 3)

Examples of training include:

- software engineering skills and practices;
- roles and responsibilities of the software engineering group and other software-related groups;
- standards, procedures, and methods for the software project;
- application domain of the software project;
- SQA objectives, procedures, and methods;
- involvement of the SQA group in the software activities;
- effective use of SQA methods and tools; and
- interpersonal communications.

Ability 4

The members of the software project receive orientation on the role, responsibilities, authority, and value of the SQA group.

Activities performed

Activity 1

A SQA plan is prepared for the software project according to a documented procedure.

This procedure typically specifies that:

1. The SQA plan is developed in the early stages of, and in parallel with, the overall project planning.
2. The SQA plan is reviewed by the affected groups and individuals.

(Activity 1)

Examples of affected groups and individuals include:

- the project software manager;
- other software managers;
- the project manager;
- customer SQA representative;
- the senior manager to whom the SQA group reports noncompliance issues; and
- the software engineering group (including all subgroups, such as software design as well as the software task leaders).

3. The SQA plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 2

The SQA group's activities are performed in accordance with the SQA plan.

The plan covers:

1. Responsibilities and authority of the SQA group.
2. Resource requirements for the SQA group (including staff, tools, and facilities).
3. Schedule and funding of the project's SQA group activities.

(Activity 2)

4. The SQA group's participation in establishing the software development plan, standards, and procedures for the project.
5. Evaluations to be performed by the SQA group.

Examples of products and activities to be evaluated include:

- operational software and support software,
- deliverable and nondeliverable products,
- software and nonsoftware products (e.g., documents),
- product development and product verification activities (e.g., executing test cases), and
- the activities followed in creating the product.

6. Audits and reviews to be conducted by the SQA group.
7. Project standards and procedures to be used as the basis for the SQA group's reviews and audits.
8. Procedures for documenting and tracking noncompliance issues to closure.

These procedures may be included as part of the plan or may be included via reference to other documents where they are contained.

9. Documentation that the SQA group is required to produce.
10. Method and frequency of providing feedback to the software engineering group and other software-related groups on SQA activities.

Activity 3

The SQA group participates in the preparation and review of the project's software development plan, standards, and procedures.

(Activity 3)

1. The SQA group provides consultation and review of the plans, standards, and procedures with regard to:
 - ☐ compliance to organizational policy,
 - ☐ compliance to externally imposed standards and requirements (e.g., standards required by the statement of work),
 - ☐ standards that are appropriate for use by the project,
 - ☐ topics that should be addressed in the software development plan, and
 - ☐ other areas as assigned by the project.
2. The SQA group verifies that plans, standards, and procedures are in place and can be used to review and audit the software project.

Activity 4

The SQA group reviews the software engineering activities to verify compliance.

1. The activities are evaluated against the software development plan and the designated software standards and procedures.

Refer to the Verifying Implementation common feature in the other key process areas for practices covering the specific reviews and audits performed by the SQA group.

2. Deviations are identified, documented, and tracked to closure.
3. Corrections are verified.

Activity 5

The SQA group audits designated software work products to verify compliance.

1. The deliverable software products are evaluated before they are delivered to the customer.
2. The software work products are evaluated against the designated software standards, procedures, and contractual requirements.

(Activity 5) 3. Deviations are identified, documented, and tracked to closure.

4. Corrections are verified.

Activity 6 The SQA group periodically reports the results of its activities to the software engineering group.

Activity 7 Deviations identified in the software activities and software work products are documented and handled according to a documented procedure.

This procedure typically specifies that:

- 1 Deviations from the software development plan and the designated project standards and procedures are documented and resolved with the appropriate software task leaders, software managers, or project manager, where possible.
2. Deviations from the software development plan and the designated project standards and procedures not resolvable with the software task leaders, software managers, or project manager are documented and presented to the senior manager designated to receive noncompliance items.
3. Noncompliance items presented to the senior manager are periodically reviewed until they are resolved.
4. The documentation of noncompliance items is managed and controlled.

Activity 8 The SQA group conducts periodic reviews of its activities and findings with the customer's SQA personnel, as appropriate.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the cost and schedule status of the SQA activities.**

Examples of measurements include:

- completions of milestones for the SQA activities compared to the plan;
- work completed, effort expended, and funds expended in the SQA activities compared to the plan; and
- numbers of product audits and activity reviews compared to the plan.

Verifying implementation

Verification 1 **The SQA activities are reviewed with senior management on a periodic basis.**

The primary purpose of periodic reviews by senior management is to provide awareness of and insight into software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2 The SQA activities are reviewed with the project manager on both a periodic and event-driven basis.

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3 Experts independent of the SQA group periodically review the activities and software work products of the project's SQA group.

Software Configuration Management

a key process area for Level 2: Repeatable

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

Software Configuration Management involves identifying the configuration of the software (i.e., selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products (e.g., the compiler).

A software baseline library is established containing the software baselines as they are developed. Changes to baselines and the release of software products built from the software baseline library are systematically controlled via the change control and configuration auditing functions of software configuration management.

This key process area covers the practices for performing the software configuration management function. The practices identifying specific

configuration items/units are contained in the key process areas that describe the development and maintenance of each configuration item/unit.

Goals

- | | |
|---------------|--|
| Goal 1 | Software configuration management activities are planned. |
| Goal 2 | Selected software work products are identified, controlled, and available. |
| Goal 3 | Changes to identified software work products are controlled. |
| Goal 4 | Affected groups and individuals are informed of the status and content of software baselines. |

Commitment to perform

- | | |
|---------------------|--|
| Commitment 1 | The project follows a written organizational policy for implementing software configuration management (SCM). |
|---------------------|--|

This policy typically specifies that:

1. Responsibility for SCM for each project is explicitly assigned.
2. SCM is implemented throughout the project's life cycle.
3. SCM is implemented for externally deliverable software products, designated internal software work products, and designated support tools used inside the project (e.g., compilers)
4. The projects establish or have access to a repository for storing configuration items/units and the associated SCM records.

(Commitment 1)

The contents of this repository are referred to as the "software baseline library" in these practices.

The tools and procedures for accessing this repository are referred to as the "configuration management library system" in these practices.

Work products that are placed under configuration management and treated as a single entity are referred to as configuration items.

Configuration items are typically decomposed into configuration components, and configuration components are typically decomposed into units. In a hardware/software system, all of the software may be considered as a single configuration item, or the software may be decomposed into multiple configuration items. In these practices the term "configuration items/units" is used to refer to the elements under configuration management.

5. The software baselines and SCM activities are audited on a periodic basis.

Ability to perform

Ability 1

A board having the authority for managing the project's software baselines (i.e., a software configuration control board - SCCB) exists or is established.

The SCCB:

1. Authorizes the establishment of software baselines and the identification of configuration items/units.
2. Represents the interests of the project manager and all groups who may be affected by changes to the software baselines.

(Ability1)

Examples of affected groups include:

- hardware quality assurance,
- hardware configuration management,
- hardware engineering,
- manufacturing engineering,
- software engineering (including all subgroups, such as software design),
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

3. Reviews and authorizes changes to the software baselines.
4. Authorizes the creation of products from the software baseline library.

Ability 2

A group that is responsible for coordinating and implementing SCM for the project (i.e., the SCM group) exists.

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time. Considerations when implementing a group include assigned tasks or activities, the size of the project, the organizational structure, and the organizational culture. Some groups, such as the software quality assurance group, are focused on project activities, and others, such as the software engineering process group, are focused on organization-wide activities.

(Ability 2)

The SCM group coordinates or implements:

1. Creation and management of the project's software baseline library.
2. Development, maintenance, and distribution of the SCM plans, standards, and procedures.
3. The identification of the set of work products to be placed under SCM.

A work product is any artifact from defining, maintaining, or using a software process.

4. Management of the access to the software baseline library.
5. Updates of the software baselines.
6. Creation of products from the software baseline library.
7. Recording of SCM actions.
8. Production and distribution of SCM reports.

Ability 3

Adequate resources and funding are provided for performing the SCM activities.

1. A manager is assigned specific responsibilities for SCM.
2. Tools to support the SCM activities are made available.

Examples of support tools include:

- workstations,
- database programs, and
- configuration management tools.

Ability 4

Members of the SCM group are trained in the objectives, procedures, and methods for performing their SCM activities.

Examples of training include:

- SCM standards, procedures, and methods; and
- SCM tools.

Ability 5

Members of the software engineering group and other software-related groups are trained to perform their SCM activities.

Examples of other software-related groups include:

- software quality assurance, and
- documentation support.

Examples of training include:

- the standards, procedures, and methods to be followed for SCM activities performed inside the software engineering group and other software-related groups; and
- the role, responsibilities, and authority of the SCM group.

Activities performed

Activity 1

A SCM plan is prepared for each software project according to a documented procedure.

This procedure typically specifies that:

1. The SCM plan is developed in the early stages of, and in parallel with, the overall project planning.

(Activity 1)

2. The SCM plan is reviewed by the affected groups.
3. The SCM plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in this key process area.

Activity 2

A documented and approved SCM plan is used as the basis for performing the SCM activities.

The plan covers:

1. The SCM activities to be performed, the schedule of activities, the assigned responsibilities, and the resources required (including staff, tools, and computer facilities).
2. The SCM requirements and activities to be performed by the software engineering group and other software-related groups.

Activity 3

A configuration management library system is established as a repository for the software baselines

This library system:

1. Supports multiple control levels of SCM.

(Activity 3)

Examples of situations leading to multiple levels of control include:

- differences in the levels of control needed at different times in the life cycle (e.g., tighter control as product matures),
- differences in the levels of control needed for software-only systems vs. systems which include both hardware and software.

2. Provides for the storage and retrieval of configuration items/units.
3. Provides for the sharing and transfer of configuration items/units between the affected groups and between control levels within the library.
4. Helps in the use of product standards for configuration items/units.
5. Provides for the storage and recovery of archive versions of configuration items/units.
6. Helps to ensure correct creation of products from the software baseline library.
7. Provides for the storage, update, and retrieval of SCM records.
8. Supports production of SCM reports.
9. Provides for the maintenance of the library structure and contents.

Examples of library maintenance functions include:

- backup/restoring of library files, and
- recovery from library errors.

Activity 4

The software work products to be placed under configuration management are identified.

(Activity 4)

1. The configuration items/units are selected based on documented criteria.

Examples of software work products that may be identified as configuration items/units include:

- process-related documentation (e.g., plans, standards, or procedures)
- software requirements,
- software design,
- software code units,
- software test procedures,
- software system build for the software test activity,
- software system build for delivery to the customer or end users,
- compilers, and
- other support tools.

2. The configuration items/units are assigned unique identifiers.
3. The characteristics of each configuration item/unit are specified.
4. The software baselines to which each configuration item/unit belongs are specified.
5. The point in its development that each configuration item/unit is placed under configuration management is specified.
6. The person responsible for each configuration item/unit (i.e., the owner, from a configuration management point of view) is identified.

Activity 5

Change requests and problem reports for all configuration items/units are initiated, recorded, reviewed, approved, and tracked according to a documented procedure.

Activity 6 **Changes to baselines are controlled according to a documented procedure.**

This procedure typically specifies that:

1. Reviews and/or regression tests are performed to ensure that changes have not caused unintended effects on the baseline.
2. Only configuration items/units that are approved by the SCCB are entered into the software baseline library.
3. Configuration items/units are checked in and out in a manner that maintains the correctness and integrity of the software baseline library.

Examples of check-in/out steps include:

- verifying that the revisions are authorized,
- creating a change log,
- maintaining a copy of the changes,
- updating the software baseline library, and
- archiving the replaced software baseline.

Activity 7 **Products from the software baseline library are created and their release is controlled according to a documented procedure.**

This procedure typically specifies that:

1. The SCCB authorizes the creation of products from the software baseline library.
2. Products from the software baseline library, for both internal and external use, are built only from configuration items/units in the software baseline library.

Activity 8 **The status of configuration items/units is recorded according to a documented procedure.**

(Activity 8)

This procedure typically specifies that:

1. The configuration management actions are recorded in sufficient detail so that the content and status of each configuration item/unit are known and previous versions can be recovered.
2. The current status and history (i.e., changes and other actions) of each configuration item/unit are maintained.

Activity 9

Standard reports documenting the SCM activities and the contents of the software baseline are developed and made available to affected groups and individuals.

Examples of reports include:

- SCCB meeting minutes,
- change request summary and status,
- trouble report summary and status (including fixes),
- summary of changes made to the software baselines,
- revision history of configuration items/units,
- software baseline status, and
- results of software baseline audits.

Activity 10

Software baseline audits are conducted according to a documented procedure.

This procedure typically specifies that:

1. There is adequate preparation for the audit.
2. The integrity of software baselines is assessed.
3. The structure and facilities of the configuration management library system are reviewed.
4. The completeness and correctness of the software baseline library contents are verified.

(Activity 10)

5. Compliance with applicable SCM standards and procedures is verified.
6. The results of the audit are reported to the project software manager.
7. Action items from the audit are tracked to closure.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the SCM activities.**

Examples of measurements include:

- number of change requests processed per unit time;
- completions of milestones for the SCM activities compared to the plan; and
- work completed, effort expended, and funds expended in the SCM activities.

Verifying implementation

Verification 1 **The SCM activities are reviewed with senior management on a periodic basis.**

The primary purpose of periodic reviews by senior management is to provide awareness of and insight into software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

(Verification 1)

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2

The SCM activities are reviewed with the project manager on both a periodic and event-driven basis.

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3

The SCM group periodically audits software baselines to verify that they conform to the documentation that defines them.

Verification 4

The software quality assurance group reviews and/or audits the activities and work products for SCM and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. Compliance with the SCM standards and procedures by:
 - ☐ the SCM group,
 - ☐ the SCCB,
 - ☐ the software engineering group, and
 - ☐ other software-related groups.
2. Occurrence of periodic software baseline audits.

Capability Maturity Model

Level 3: Defined

The software process for both management and engineering activities is documented, standardized, and integrated into an organization-wide software process. All projects use a documented and approved version of the organization's process for developing and maintaining software.

Key process areas

Organization Process Focus	L3-1
Organization Process Definition	L3-11
Training Program	L3-25
Integrated Software Management	L3-37
Software Product Engineering	L3-59
Intergroup Coordination	L3-83
Peer Reviews	L3-93

Organization Process Focus

a key process area for Level 3: Defined

The purpose of Organization Process Focus is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability.

Organization Process Focus involves developing and maintaining an understanding of the organization's and projects' software processes and coordinating the activities to assess, develop, maintain, and improve these processes.

The organization provides the long-term commitments and resources to coordinate the development and maintenance of the software processes across current and future software projects via a group such as a software engineering process group. This group is responsible for the organization's software process activities. It is specifically responsible for the development and maintenance of the organization's standard software process and related process assets (as described in the Organization Process Definition key process area), and it coordinates the process activities with the software projects.

Goals

Goal 1	Software process development and improvement activities are coordinated across the organization.
--------	--

- Goal 2** **The strengths and weaknesses of the software processes used are identified relative to a process standard.**
- Goal 3** **Organization-level process development and improvement activities are planned.**

Commitment to perform

- Commitment 1** **The organization follows a written organizational policy for coordinating software process development and improvement activities across the organization.**

This policy typically specifies that:

1. A group is established that is responsible for the organization-level software process activities and coordinating these activities with the projects.
2. The software processes used by the projects are assessed periodically to determine their strengths and weaknesses.
3. The software processes used by the projects are appropriately tailored from the organization's standard software process.

Refer to Activity 1 of the Integrated Software Management key process area for practices covering tailoring of the organization's standard software process.

4. Improvements to, and other useful information on, each project's software process, tools, and methods are available to other projects.

- Commitment 2** **Senior management sponsors the organization's activities for software process development and improvement.**

(Commitment 2) Senior management:

1. Demonstrates to the organization's staff and managers its commitment to these software process activities.
2. Establishes long-term plans and commitments for funding, staffing, and other resources.
3. Establishes strategies for managing and implementing the activities for process development and improvement.

Commitment 3 Senior management oversees the organization's activities for software process development and improvement.

Senior management:

1. Ensures that the organization's standard software process supports its business goals and strategies.
2. Advises on setting priorities for software process development and improvement.
3. Participates in establishing plans for software process development and improvement.
 - ☐ Senior management coordinates software process requirements and issues with higher level staff and managers.
 - ☐ Senior management coordinates with the organization's managers to secure the managers' and staff's support and participation.

Ability to perform

Ability 1

A group that is responsible for the organization's software process activities exists.

(Ability 1)

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time. Considerations when implementing a group include assigned tasks or activities, the size of the project, the organizational structure, and the organizational culture. Some groups, such as the software quality assurance group, are focused on project activities, and others, such as the software engineering process group, are focused on organization-wide activities.

1. Where possible, this group is staffed by a core of software technical professionals who are assigned full time to the group, possibly supported by others, on a part-time basis.

The most common example of this group is a software engineering process group (SEPG).

2. This group is staffed to represent the software engineering discipline and software-related disciplines.

Examples of software engineering and software-related disciplines include:

- software requirements analysis,
- software design,
- coding,
- software test,
- software configuration management, and
- software quality assurance.

Ability 2

Adequate resources and funding are provided for the organization's software process activities.

(Ability 2)

1. Experienced individuals who have expertise in specialized areas are committed to support this group.

Examples of specialized areas include:

- software reuse,
- computer-aided software engineering (CASE) technology,
- measurement, and
- training course development.

2. Tools to support the organization's software process activities are made available.

Examples of support tools include:

- statistical analysis tools,
- desktop publishing tools,
- database management systems, and
- process modeling tools.

Ability 3

Members of the group responsible for the organization's software process activities receive required training to perform these activities.

Examples of training include:

- software engineering practices;
- process control techniques;
- organization change management;
- planning, managing, and monitoring the software process; and
- technology transition.

(Ability 3)

Refer to the Training Program key process area.

Ability 4

Members of the software engineering group and other software-related groups receive orientation on the organization's software process activities and their roles in those activities.

Refer to the Training Program key process area.

Activities performed

Activity 1

The software process is assessed periodically, and action plans are developed to address the assessment findings.

Assessments are typically conducted every 1-1/2 to 3 years.

Assessments look at all software processes used in the organization, but may do this by sampling process areas and projects.

An example of a method to assess an organization's software process capability is the SEI Software Process Assessment method.

The action plan identifies:

- which assessment findings will be addressed,
- guidelines for implementing the changes to address findings, and
- the groups or individuals responsible for implementing the changes.

Activity 2

The organization develops and maintains a plan for its software process development and improvement activities.

This plan:

1. Uses the action plans from the software process assessments and other organization improvement initiatives as primary inputs.
2. Defines the activities to be performed and the schedule for these activities.
3. Specifies the groups and individuals responsible for the activities.
4. Identifies the resources required, including staff and tools.
5. Undergoes peer review when initially released and whenever major revisions are made.

Refer to the Peer Reviews key process area.

6. Is reviewed and agreed to by the organization's software managers and senior managers.

Activity 3

The organization's and projects' activities for developing and improving their software processes are coordinated at the organization level.

This coordination covers the development and improvement of:

1. The organization's standard software process.

Refer to Activities 1 and 2 of the Organization Process Definition key process area for practices covering the organization's standard software process.

(Activity 3)**2. The projects' defined software processes**

Refer to Activities 1 and 2 of the Integrated Software Management key process area for practices covering the project's defined software process.

Activity 4**The use of the organization's software process database is coordinated at the organizational level.**

The organization's software process database is used to collect information on the software processes and resulting software products of the organization and the projects.

Refer to Activity 5 of the Organization Process Definition key process area for practices covering the organization's software process database.

Activity 5**New processes, methods, and tools in limited use in the organization are monitored, evaluated, and, where appropriate, transferred to other parts of the organization.****Activity 6****Training for the organization's and projects' software processes is coordinated across the organization.**

1. Plans for training on subjects related to the organization's and projects' software processes are prepared.
2. Where appropriate, training may be prepared and conducted by the group responsible for the organization's software process activities (e.g., software engineering process group) or by the training group.

(Activity 6)

Refer to the Training Program key process area.

Activity 7

The groups involved in implementing the software processes are informed of the organization's and projects' activities for software process development and improvement.

Examples of means to inform and involve these people include:

- electronic bulletin boards on process,
- process advisory boards,
- working groups,
- information exchange meetings,
- surveys,
- process improvement teams, and
- informal discussions.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the organization's process development and improvement activities.

Examples of measurements include:

- work completed, effort expended, and funds expended in the organization's activities for process assessment, development, and improvement compared to the plans for these activities; and
- results of each software process assessment, compared to the results and recommendations of previous assessments.

Verifying implementation

Verification 1

The activities for software process development and improvement are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

1. Progress and status of the activities to develop and improve the software process are reviewed against the plan.
2. Conflicts and issues not resolved at lower levels are addressed.
3. Action items are assigned, reviewed, and tracked to closure.
4. A summary report from each review is prepared and distributed to the affected groups and individuals.

Organization Process Definition

a key process area for Level 3: Defined

The purpose of Organization Process Definition is to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization.

Organization Process Definition involves developing and maintaining the organization's standard software process, along with related process assets, such as descriptions of software life cycles, process tailoring guidelines and criteria, the organization's software process database, and a library of software process-related documentation.

These assets may be collected in many ways, depending on the organization's implementation of Organization Process Definition. For example, the descriptions of the software life cycles may be an integral part of the organization's standard software process or parts of the library of software process-related documentation may be stored in the organization's software process database.

The organization's software process assets are available for use in developing, implementing, and maintaining the projects' defined software processes. (The practices related to the development and maintenance of the project's defined software process are described in the Integrated Software Management key process area.)

Goals

- Goal 1** A standard software process for the organization is developed and maintained.
- Goal 2** Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

Commitment to perform

- Commitment 1** The organization follows a written policy for developing and maintaining a standard software process and related process assets.

The organization's software process assets include:

- the organization's standard software process,
- guidelines and criteria for the projects' tailoring of the organization's standard software process,
- descriptions of software life cycles approved for use,
- the organization's software process database, and
- a library of software process-related documentation previously developed and available for reuse.

This policy typically specifies that:

1. A standard software process is defined for the organization.

(Commitment 1)

The primary purposes of a standard software process are to maximize the sharing of process assets and experiences across the projects and to provide the ability to define and aggregate a standard set of process measurements from the projects at the organization level.

The organization's standard software process may contain multiple software processes. Multiple software processes may be needed to address the needs of different applications, life cycles, methodologies, and tools, which the software projects may compose in multiple ways.

2. A project's defined software process is a tailored version of the organization's standard software process.

Refer to Activity 1 of the Integrated Software Management key process area for practices covering tailoring of the organization's standard software process.

3. The organization's software process assets are maintained.
4. Information collected from the projects is organized and used to improve the organization's standard software process.

Examples of collected information include:

- process and product measurements,
- lessons learned, and
- other process-related documentation.

Ability to perform

Ability 1

Adequate resources and funding are provided for developing and maintaining the organization's standard software process and related process assets.

1. The development and maintenance of the organization's standard software process and related process assets is performed or coordinated by the group responsible for the organization's software process activities (e.g., software engineering process group).

Refer to the Organization Process Focus key process area for practices covering the group responsible for the organization's software process activities.

2. Tools to support process development and maintenance are made available.

Examples of support tools include:

- desktop publishing tools,
- database management systems, and
- process modeling tools.

Ability 2

The individuals who develop and maintain the organization's standard software process and related process assets receive required training to perform these activities.

Examples of training include:

- software engineering practices and methods,
- process analysis and documentation methods, and
- process modeling.

(Ability 2)

Refer to the Training Program key process area.

Activities performed

Activity 1

The organization's standard software process is developed and maintained according to a documented procedure.

This procedure typically specifies that:

1. The organization's standard software process satisfies the software policies, process standards, and product standards imposed on the organization, as appropriate.
2. The organization's standard software process satisfies the software process and product standards that are commonly imposed on the organization's projects by their customers, as appropriate.
3. State-of-the-practice software engineering tools and methods are incorporated into the organization's standard software process, as appropriate.
4. The internal process interfaces between the software disciplines are described.

Examples of software engineering disciplines include:

- software requirements analysis,
- software design,
- coding,
- software testing,
- software configuration management, and
- software quality assurance.

(Activity 1)

5. The external process interfaces between the software process and the processes of other affected groups are described.

Examples of other affected groups include:

- system engineering,
- system test,
- contract management, and
- documentation support.

6. Changes proposed for the organization's standard software process are documented, reviewed, and approved by the group responsible for the organization's software process activities (e.g., software engineering process group) before they are incorporated.

Examples of sources for change include:

- the findings and recommendations of software process assessments,
- results of the project's tailoring of the organization's standard software process,
- lessons learned from monitoring the organization's and projects' software process activities,
- changes proposed by the organization's staff and managers, and
- process and product measurement data that are analyzed and interpreted.

7. Plans for introducing changes to the software process of ongoing projects are defined as appropriate.
8. The description of the organization's standard software process undergoes peer review when initially developed and whenever significant changes or additions are made.

(Activity 1)

Refer to the Peer Reviews key process area.

9. The description of the organization's standard software process is placed under configuration management.

Refer to the Software Configuration Management key process area.

Activity 2

The organization's standard software process is documented according to established organization standards.

These standards typically specify that:

1. The process is decomposed into constituent process elements to the granularity needed to understand and describe the process.

Each process element covers a well-defined, bounded, closely related set of activities.

Examples of process elements include:

- software estimating element,
- software design element,
- coding element, and
- peer review element.

The descriptions of the process elements may be templates to be filled in, fragments to be completed, abstractions to be refined, or complete descriptions to be modified or used unmodified.

2. Each process element is described and addresses:
 - ☐ the required procedures, practices, methods, and technologies;
 - ☐ the applicable process and product standards;

(Activity 2)

- ☐ the responsibilities for implementing the process;
- ☐ the required tools and resources;
- ☐ inputs;
- ☐ the software work products produced;
- ☐ the software work products that should undergo peer review;
- ☐ the readiness and completion criteria; and
- ☐ the product and process data to be collected.

3. The relationships of the process elements are described and address:

- ☐ the ordering,
- ☐ the interfaces, and
- ☐ the interdependencies.

This relationship of the process elements is sometimes referred to as a software process architecture.

Activity 3

Descriptions of software life cycles that are approved for use by the projects are documented and maintained.

Examples of software life cycles include:

- waterfall,
- overlapping waterfall,
- spiral,
- serial build, and
- single prototype/overlapping waterfall.

1. The software life cycles are compatible with the organization's standard software process.
2. Changes proposed for the descriptions of software life cycles are documented, reviewed, and approved by the group responsible for the organization's software process activities (e.g., software engineering process group) before they are incorporated.

(Activity 3)

3. The descriptions of the software life cycles undergo peer review when initially documented and whenever significant changes or additions are made.

Refer to the Peer Reviews key process area.

4. The descriptions of the software life cycles are managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 4

Guidelines and criteria for the projects' tailoring of the organization's standard software process are developed and maintained.

1. The tailoring guidelines and criteria cover:
- ☐ selecting and tailoring the software life cycle for the project,
 - ☐ tailoring the organization's standard software process to accommodate the software life cycle and the project's characteristics, and

(Activity 4)

Examples of tailoring include:

- adapting the process for a new product line or host environment,
- customizing the process for a specific project or class of projects, and
- elaborating and adding detail to the process so that the resulting project's defined software process can be enacted.

- ☐ standards for documenting the project's defined software process.
2. Changes proposed for the tailoring guidelines and criteria are documented, reviewed, and approved by the group responsible for the organization's software process activities (e.g., software engineering process group) before they are incorporated.
 3. The tailoring guidelines and criteria are managed and controlled.

Activity 5

The organization's software process database is established and maintained.

1. The database is established to collect and make available data on the software processes and resulting software work products.

Examples of process and work product data include:

- estimates of software size, effort, and cost;
- actual data on software size, effort, and cost;
- productivity data;
- quality measurements;
- peer review coverage and efficiency;
- test coverage and efficiency;
- software reliability measures;
- number and severity of defects found in the software requirements; and
- number and severity of defects found in the software code.

(Activity 5)

2. The data entered into the database is reviewed to ensure the integrity of the database contents.

In addition, the database also contains or references the actual measurement data and related information and data needed to understand and interpret the measurement data and assess it for reasonableness and applicability.

3. The database is managed and controlled.
4. User access to the database contents is controlled to ensure completeness, integrity, and accuracy of the data.

Access is limited to those who have a need to enter, change, view, analyze, or extract data.

Sensitive data are protected and access to these data is appropriately controlled.

Activity 6

A library of software process-related documentation is established and maintained.

Examples of software process-related documentation include:

- the description of a project's defined software process,
- a project's standards,
- a project's procedures,
- a project's software development plans,
- a project's measurement plans, and
- a project's process training materials.

1. Candidate documentation items are reviewed and appropriate items that may be useful in the future are included in the library.

(Activity 6)

2. The documentation items are catalogued for easy access.
3. Revisions made to documentation items currently in the library are reviewed, and the library contents are updated as appropriate.
4. The library contents are made available for use by the software projects and other software-related groups.

Examples of software-related groups include:

- software quality assurance
- software configuration management,
- software test, and
- documentation support.

5. The use of each documentation item is reviewed periodically, and the results are used to maintain the library contents.
6. The library contents are managed and controlled.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the organization's process definition activities.**

Examples of measurements include:

- status of the schedule milestones for process development and maintenance, and
- costs for the process definition activities.

Verifying implementation

Verification 1

The software quality assurance group reviews and/or audits the organization's activities and work products for developing and maintaining the organization's standard software process and related process assets and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, these reviews and/or audits verify that:

1. The appropriate standards are followed in developing, documenting, and maintaining the organization's standard software process and related process assets.
2. The organization's standard software process and related process assets are controlled and used appropriately.

Training Program

a key process area for Level 3: Defined

The purpose of the Training Program key process area is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently.

Training Program involves first identifying the training needed by the organization, projects, and individuals, then developing or procuring training to address the identified needs.

Each software project evaluates its current and future skill needs and determines how these skills will be obtained. Some skills are effectively and efficiently imparted through informal vehicles (e.g., on-the-job training and informal mentoring), whereas other skills need more formal training vehicles (e.g., classroom training and guided self-study) to be effectively and efficiently imparted. The appropriate vehicles are selected and used.

This key process area covers the practices for the group performing the training function. The practices identifying the specific training topics (i.e., knowledge or skill needed) are contained in the Ability to Perform common feature of the individual key process areas.

Goals

- | | |
|--------|---|
| Goal 1 | Training activities are planned. |
| Goal 2 | Training for developing the skills and knowledge needed to perform software management and technical roles is provided. |

- Goal 3** **Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.**

Commitment to perform

- Commitment 1** **The organization follows a written policy for meeting its training needs.**

This policy typically specifies that:

1. The needed skills and knowledge for each software management and technical role are identified.
2. Training vehicles for imparting skills and knowledge are identified and approved.

Examples of approved training vehicles include:

- classroom training,
- computer-aided instruction,
- guided self-study,
- formal apprenticeship and mentoring programs, and
- facilitated videos.

3. Training is provided to build the skill base of the organization, to fill the specific needs of the projects, and to develop the skills of individuals.
4. Training is developed within the organization or obtained from outside the organization when appropriate.

(Commitment 1)

Examples of external sources of training include:

- customer-provided training,
- commercially available training courses,
- academic programs,
- professional conferences, and
- seminars.

Ability to perform

Ability 1

A group responsible for fulfilling the training needs of the organization exists.

The members of the training group may include full-time or part-time instructors drawn from the organization; the members may also be drawn from external sources.

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time. Considerations when implementing a group include assigned tasks or activities, the size of the project, the organizational structure, and the organizational culture. Some groups, such as the software quality assurance group, are focused on project activities, and others, such as the software engineering process group, are focused on organization-wide activities.

Ability 2

Adequate resources and funding are provided for implementing the training program.

(Ability 2)

Examples of training program elements include:

- the organization's training plan,
- training materials,
- development or procurement of training,
- conduct of training,
- training facilities,
- evaluation of training, and
- maintaining records of training.

1. A manager is designated to be responsible for implementing the organization's training program.
2. Tools to support the training program activities are made available.

Examples of support tools include:

- workstations,
- instructional design tools,
- database programs, and
- packages for developing presentation materials.

3. Appropriate facilities are made available to conduct training.

Classroom training facilities should be separated from the students' work environment to eliminate interruptions.

Where appropriate, training is conducted in settings that closely resemble actual performance conditions and includes activities to simulate actual work situations.

Ability 3

Members of the training group have the necessary skills and knowledge to perform their training activities.

(Ability 3)

Examples of ways to provide these skills and knowledge include:

- training in instructional techniques, and
- refresher training in the subject matter.

Ability 4

Software managers receive orientation on the training program.

Activities performed

Activity 1

Each software project develops and maintains a training plan that specifies its training needs.

The plan covers:

1. The set of skills needed and when those skills are needed.
2. The skills for which training is required and the skills that will be obtained via other vehicles.

Some skills are effectively and efficiently imparted through informal vehicles (e.g., informal training and presentations, reading books and journals, "chalk talks," brown-bag lunch seminars, on-the-job training, and informal mentoring); while other skills, to be effectively and efficiently imparted, need to be based on more formal training vehicles (e.g., classroom training, computer-aided instructions, guided self-study, facilitated video, and formal apprenticeship and mentoring programs).

3. The training that is required, for whom it is required, and when it is required.

(Activity 1)

Refer to the Ability to Perform common feature in all other key process areas for examples of specific training needs.

Where appropriate, training for individuals is tied to their work responsibilities so that on-the-job activities or other outside experiences will reinforce the training within a reasonable time after the training.

4. How training will be provided.

Training may be provided by the software project, by the organization's training group, or by an external organization.

Examples of training appropriately done by the software project include:

- training in specific applications and requirements of the project,
- training in the project's software architecture, and
- other training more effectively or efficiently performed at the project level.

Activity 2

The organization's training plan is developed and revised according to a documented procedure.

This procedure typically specifies that:

1. The plan uses the software projects' training needs identified in their training plans.
2. The specific training to be provided is identified based on the skills needed by the organization and when those skills are needed.

(Activity 2)

3. The organization's training plan is revised, as appropriate, to incorporate changes.
4. The organization's training plan is reviewed by the affected individuals when it is initially released and whenever major revisions are made.

Examples of affected individuals include:

- senior management,
- software managers, and
- managers of software-related groups.

5. The organization's training plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

6. The organization's training plan is readily available to the affected groups and individuals.

(Activity 2)

Examples of affected groups and individuals include:

- senior management,
- the training group,
- the managers of software-related groups,
- software engineering (including all subgroups, such as software design),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

Activity 3

The training for the organization is performed in accordance with the organization's training plan.

The plan covers:

1. The specific training needed within the organization and when it is needed.
2. The training that will be obtained from external sources and training that will be provided by the training group.
3. The funding and resources (including staff, tools, and facilities) needed to prepare and conduct or procure the training.
4. Standards for instructional materials used in training courses developed by the training group.
5. The schedule for developing and revising the training courses that will be developed by the training group.
6. The schedule for conducting the training, based on the projected need dates and the projected number of students.

(Activity 3)

7. The procedures for:

- ☐ selecting the individuals who will receive the training,
- ☐ registering and participating in the training,
- ☐ maintaining records of the training provided, and
- ☐ collecting, reviewing, and using training evaluations and other training feedback.

Activity 4

Training courses prepared at the organization level are developed and maintained according to organization standards.

These standards require that:

1. A description of each training course is developed.

Examples of the topics addressed by the description include:

- intended audience,
- preparation for participating,
- training objectives,
- length of the training,
- lesson plans,
- criteria for determining the students' satisfactory completion,
- procedures for periodically evaluating the effectiveness of the training, and
- special considerations, such as piloting and field testing the training course, needs for refresher training, and opportunities for follow-up training.

2. The materials for the training course are reviewed.

(Activity 4)

Examples of individuals who review the training materials include:

- instructional experts,
- subject matter experts, and
- representative students from pilot sessions of the training course being reviewed.

3. The materials for the training courses are managed and controlled.

Activity 5

A waiver procedure for required training is established and used to determine whether individuals already possess the knowledge and skills required to perform in their designated roles.

Activity 6

Records of training are maintained.

1. Records are kept of all students who successfully complete each training course or other approved training activity.
2. Records are kept of all students who successfully complete their designated required training.
3. Records of successfully completed training are made available for consideration in assignments of the staff and managers.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the training program activities.

(Measurement 1)

Examples of measurements include:

- actual attendance at each training course compared to the projected attendance,
- progress in providing training courses compared to the organization's and projects' training plans, and
- number of training waivers approved over time.

Measurement 2 **Measurements are made and used to determine the quality of the training program.**

Examples of measurements include:

- results of post-training tests,
- reviews of the courses from the students, and
- feedback from the software managers.

Verifying implementation

Verification 1 **The training program activities are reviewed with senior management on a periodic basis.**

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

(Verification 1)

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2

The training program is independently evaluated on a periodic basis for consistency with, and relevance to, the organization's needs.

Verification 3

The training program activities and work products are reviewed and/or audited and the results are reported.

At a minimum, the reviews and/or audits verify that:

1. The process for developing and revising the organization's training plan is followed.
2. The process for developing and revising a training course is followed.
3. Training records are properly maintained.
4. Individuals designated as requiring specific training complete that training.
5. The organization's training plan is followed.

Integrated Software Management

a key process area for Level 3: Defined

The purpose of Integrated Software Management is to integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets, which are described in Organization Process Definition.

Integrated Software Management involves developing the project's defined software process and managing the software project using this defined software process. The project's defined software process is tailored from the organization's standard software process to address the specific characteristics of the project.

The software development plan is based on the project's defined software process and describes how the activities of the project's defined software process will be implemented and managed. The management of the software project's size, effort, cost, schedule, staffing, and other resources is tied to the tasks of the project's defined software process.

Since the projects' defined software processes are all tailored from the organization's standard software process, the software projects can share process data and lessons learned.

The basic practices for estimating, planning, and tracking a software project are described in the Software Project Planning and Software

Project Tracking and Oversight key process areas. They focus on recognizing problems when they occur and adjusting the plans and/or performance to address the problems. The practices of this key process area build on, and are in addition to, the practices of those two key process areas. The emphasis of Integrated Software Management shifts to anticipating problems and acting to prevent or minimize the effects of these problems.

Goals

- Goal 1** The project's defined software process is a tailored version of the organization's standard software process.
- Goal 2** The project is planned and managed according to the project's defined software process.

Commitment to perform

- Commitment 1** The project follows a written organizational policy requiring that the software project be planned and managed using the organization's standard software process and related process assets.

Refer to the Organization Process Definition key process area for practices covering the organization's standard software process and related process assets.

(Commitment 1) This policy typically specifies that:

1. Each project documents the project's defined software process by tailoring the organization's standard software process.
2. The project's deviations from the organization's standard software process are documented and approved.
3. Each project performs its software activities in accordance with the project's defined software process.
4. Each project collects and stores appropriate project measurement data in the organization's software process database.

Refer to Activity 5 of the Organization Process Definition key process area for practices covering the organization's software process database.

Ability to perform

Ability 1

Adequate resources and funding are provided for managing the software project using the project's defined software process.

Refer to Ability 3 of the Software Project Planning key process area and Ability 3 of the Software Project Tracking and Oversight key process area for practices covering resources and funding for software project planning, tracking, and oversight.

Ability 2

The individuals responsible for developing the project's defined software process receive required training in how to tailor the organization's standard software process and use the related process assets.

(Ability 2)

Examples of training include:

- using the software process database,
- using the organization's standard software process, and
- using the guidelines and criteria for tailoring the organization's standard software process to meet the needs of the software project.

Refer to the Training Program key process areas.

Ability 3

The software managers receive required training in managing the technical, administrative, and personnel aspects of the software project based on the project's defined software process.

Refer to Ability 4 of the Software Project Planning key process area and Ability 4 of the Software Project Tracking and Oversight key process area for practices covering training for software project planning, tracking, and oversight.

Examples of training include:

- methods and procedures for software estimating, planning, and tracking based on the project's defined software process; and
- methods and procedures for identifying, managing, and communicating software risks.

Refer to the Training Program key process area.

Activities performed

Activity 1

The project's defined software process is developed by tailoring the organization's standard software process according to a documented procedure.

Refer to Activity 2 of Organization Process Definition key process area for practices covering the contents of the organization's standard software process.

This procedure typically specifies that:

1. A software life cycle is:

- ☐ selected from among those approved by the organization, to satisfy the project's contractual and operational constraints;

Refer to Activity 3 of the Organization Process Definition key process area for practices covering approved software life cycles.

- ☐ modified, if necessary, in ways permitted by the organization's tailoring guidelines and criteria; and
- ☐ documented according to the organization's standards.

Refer to Activity 4 of the Organization Process Definition key process area for practices covering the organization's tailoring guidelines and criteria.

2. The description of the project's defined software process is documented.

(Activity 1)

Refer to Activity 2 of the Organization Process Definition key process area for practices covering the expected contents of a process definition.

The tailoring uses the organization's process assets as appropriate.

3. Tailoring of the organization's standard software process for the project is reviewed by the group responsible for coordinating the organization's software process activities (e.g., software engineering process group) and approved by senior management.

Refer to Activity 6 of the Organization Process Definition key process area for practices covering the library of software process-related documentation.

- ☐ Waivers for deviations from the organization's standard software process are documented and are reviewed and approved by senior management.
4. Waivers for deviations from contractual software process requirements are documented and are reviewed and approved by senior management and the software project's customer, as appropriate.
5. The description of the project's defined software process is managed and controlled.

(Activity 1)

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 2

Each project's defined software process is revised according to a documented procedure.

This procedure typically specifies that:

1. Changes derived from the following are documented and systematically reviewed:
 - ☐ lessons learned from monitoring the software activities of the organization's projects,
 - ☐ changes proposed by the software project, and
 - ☐ process and work product measurement data.
2. Changes to the project's defined software process are reviewed and approved before they are incorporated.

Examples of individuals who review the changes include:

- members of the groups responsible for the organization's software process activities (e.g., software engineering process group),
- the software managers, and
- the project software manager.

(Activity 2)

Examples of individuals who approve the changes include:

- the project software manager, and
- the project manager.

Activity 3

The project's software development plan, which describes the use of the project's defined software process, is developed and revised according to a documented procedure.

Refer to Activities 6 and 7 of the Software Project Planning key process area and Activities 1 and 2 of the Software Project Tracking and Oversight key process area for practices covering the software development plan.

Activity 4

The software project is managed in accordance with the project's defined software process.

Refer to the Software Project Planning and the Software Project Tracking and Oversight key process areas for basic practices covering managing a software project.

The project's defined software process typically specifies that:

1. Provisions are made for gathering, analyzing, and reporting measurement data needed to manage the software project.
2. The activities for software estimating, planning, and tracking are tied to the key tasks and work products of the project's defined software process.
3. Readiness and completion criteria are established, documented, and used to authorize initiation and determine completion of key tasks.

(Activity 4)

4. Documented criteria are defined to indicate when to replan the software project.
5. Technical and management lessons learned are documented and stored in the organization's library of software process-related documentation.

Refer to Activity 6 of the Organization Process Definition key process area for practices covering the organization's library of software process-related documentation.

6. Technical and management lessons learned from monitoring the activities of other projects in the organization are systematically reviewed and used to estimate, plan, track, and replan the software project.
7. The staffing plan addresses the software project's needs for individuals with special skills and application domain knowledge.
8. Training needs are identified and documented to fit the specific needs of the software project.

Refer to Activity 1 of the Training Program key process area for practices covering the identification of the project's training needs.

9. The software plans and processes followed in interacting with other groups are adjusted to account for disparities with these groups and for other potential problems.

(Activity 4)

Examples of disparities and problems include:

- differences in process maturity,
- process incompatibility, and
- various business factors.

Activity 5

The organization's software process database is used for software planning and estimating.

Refer to Activity 5 of the Organization Process Definition key process area for practices covering the organization's software process database.

1. The database is used as a source of data to estimate, plan, track, and replan a software project; data for similar software projects are used when possible.

Examples of data contained in the organization's software process database include:

- size of the software work products,
- software effort,
- software cost,
- schedule,
- staffing, and
- technical activities.

2. Parameter values used to derive estimates for software size, effort, cost, schedule, and use of critical computer resources are compared to those of other software projects to assess their validity.
 - ☐ Similarities and differences to the other projects in terms of application domain and design approach are assessed and recorded.
 - ☐ Rationales for similarities and differences between the parameter values are recorded.

(Activity 5)

- ☐ The reasoning used to judge the credibility of the project's estimates is recorded.
- 3. The software project provides appropriate software planning data, replanning data, and actual measured data for storage in the organization's software process database.

Examples of data recorded by the software project include:

- the task description,
- the assumptions,
- the estimates,
- the revised estimates,
- the actual measured data, and
- the associated information needed to reconstruct the estimates, assess their reasonableness, and derive estimates for new work.

Activity 6

The size of the software work products (or size of changes to the software work products) is managed according to a documented procedure.

Refer to Activity 9 of the Software Project Planning key process area and Activity 5 of the Software Project Tracking and Oversight key process area for basic practices covering planning and tracking size of software work products.

This procedure typically specifies that:

1. A group that is independent of the software engineering group reviews the procedures for estimating the size of the software work products, and provides guidance in using historical data from the organization's software process database to establish credible estimates.

(Activity 6)

An example of an independent group is a software estimating group.

An example of a method to evaluate the credibility of software size estimates is a function-by-function comparison to a completed system.

- ☐ The individuals who prepare the size estimates ensure that the procedures and data used in the estimates are appropriate.
 - ☐ When the validity of a size estimate is questioned, a team of peers and experts reviews the estimate.
2. A contingency factor is applied to the size estimate for each software element identified as a software risk.
 - ☐ The rationale for the contingency is documented.
 - ☐ The risks associated with reducing or eliminating the contingency are assessed and documented.
 3. Off-the-shelf or reusable software components are identified.
 - ☐ Reuse measurements account for the reuse of requirements, design, code, test plan, and test procedures, etc.
 - ☐ The effort to modify and incorporate reusable components is factored into the size estimates.
 4. Factors which could significantly affect the size of the software work products are identified and monitored closely.
 5. A size threshold is established for each managed software element which, when projected to be exceeded, requires action.

Activity 7

The project's software effort and costs are managed according to a documented procedure.

(Activity 7)

Refer to Activity 10 of the Software Project Planning key process area and Activity 6 of the Software Project Tracking and Oversight key process area for basic practices covering planning and tracking software efforts and costs.

This procedure typically specifies that:

1. Software effort, cost, and staffing profile models, if used, are adapted to the project and use available historical data where appropriate.
2. Referenced productivity and cost data are adjusted to incorporate project variables.

Examples of project variables include:

- the geographic locations of the project's groups and organizations (e.g., subcontractor),
- the size and complexity of the system,
- the stability of the requirements,
- the host environment for development,
- the target environment of the system,
- the developers' familiarity and experience with the application,
- the availability of resources, and
- other special constraints.

3. The overall software effort and cost is allocated to individually managed tasks or stages as needed to manage the effort and cost effectively.
4. When the software effort and cost status is reviewed and the estimates are revised, actual expenditures over time and against work completed are compared to the software development plan and used to refine the effort and cost estimates for remaining work.

(Activity 7)

- ☐ Parameter values of the models used in estimating software effort and costs are updated whenever major changes are made to the software requirements.
 - ☐ Actual data on project productivity and other new software costs are used where appropriate.
5. An effort and cost threshold is established for each individually managed software task or stage which, when projected to be exceeded, requires action.

Activity 8

The project's critical computer resources are managed according to a documented procedure.

Refer to Activity 11 of the Software Project Planning key process area and Activity 7 of the Software Project Tracking and Oversight key process area for basic practices covering planning and tracking critical computer resources.

This procedure typically specifies that:

1. Estimates for the project's critical computer resources are derived based on historical experience, simulations, prototyping, or analysis, as appropriate.
 - ☐ Sources and rationale for estimates are documented.
 - ☐ Similarities and differences between the project and the sources for historical data in terms of application domain and design approach are assessed and recorded.
 - ☐ The reasoning used to judge the credibility of the estimates is recorded.
2. The planned computer resources, the system requirements allocated to software, the software requirements, and/or the software design are adjusted to achieve the project's critical computer resource requirements.
3. The available computer resources are allocated to the software components.

(Activity 8)

4. The available capacity for the critical computer resources provides for a specified reserve capacity when the initial estimates are made.
5. A threshold is established for each critical computer resource which, when projected to be exceeded, requires action.

Activity 9

The critical dependencies and critical paths of the project's software schedule are managed according to a documented procedure.

Refer to Activity 12 of the Software Project Planning key process area, Activity 8 of the Software Project Tracking and Oversight key process area, and Activity 4 of the Intergroup Coordination key process area for practices covering negotiating and tracking critical dependencies.

This procedure typically specifies that:

1. Milestones, tasks, commitments, critical dependencies, staffing, costs, and reviews are allocated in the schedule consistent with the project's defined software process.
 - ☐ The software schedule identifies specific tasks and milestones whose completion can be objectively determined (i.e., a binary or yes/no determination).

Different levels of schedule detail, appropriately tied to each other, are developed to accommodate the needs of different groups and individuals.

2. Critical dependencies are defined, negotiated, and reflected in the software schedule.

(Activity 9)

Critical dependencies include both those within the software engineering group (i.e., between subgroups) and between the software engineering group and other affected groups.

3. Schedule critical paths are defined and reflected in the software schedule.
4. The software project's critical dependencies and schedule critical paths are tracked on a regular basis.
5. Specific documented threshold criteria are established for each critical path which, when projected to be exceeded, require action.

Examples of actions include:

- conducting analyses and simulations to tradeoff function, quality, cost, schedule, staffing, and other resources;
- allocating contingencies and schedule slack, if available;
- evaluating the effects of contemplated actions on all critical paths; and
- making decisions visible to the affected groups.

Activity 10

The project's software risks are identified, assessed, documented, and managed according to a documented procedure.

Refer to Activity 13 of the Software Project Planning key process area and Activity 10 of the Software Project Tracking and Oversight key process area for basic practices covering identifying and tracking risk.

(Activity 10)

Examples of software risks that are to be managed include significant possibilities that the software project could fail to meet its objectives in areas such as:

- schedule,
- cost,
- functionality,
- throughput or real-time performance,
- reliability or availability, and
- use of critical computer resources.

Examples of activities to manage risks include:

- early identification of high-risk project objectives;
- identification of events that could introduce or increase risks;
- prototyping or early implementation of high-risk modules;
and
- close monitoring of key project risk indicators.

This procedure typically specifies that:

1. A software risk management plan is documented and used to identify and manage the software risks.

(Activity 10)

Examples of items in a software risk management plan include:

- resources required (including staff and tools);
- risk management methods (e.g., identification, analysis, prioritization, planning, monitoring, and resolution);
- list of identified risks (including assessment, prioritization, status, and plans);
- risk management schedule;
- responsibilities and authorities;
- method and frequency of communicating risk status and activities; and
- measurements.

2. Contingency planning is based on the project's defined software process and is performed throughout the project's software life cycle.

Examples of areas covered by contingency planning activities include:

- identification of options,
- impact assessment of options,
- technical feasibility of options,
- allocation of management reserves, and
- decision criteria on when to pursue an option.

3. Alternatives for each software risk are defined, where possible, along with criteria for selecting among the alternatives.
4. The initial release and major revisions to the software risk management plan undergo peer review.

Refer to the Peer Reviews key process area.

5. The software risk management plan is managed and controlled.

(Activity 10)

6. Software risks are tracked, reassessed, and replanned at selected project milestones, at designated risk checkpoints, and during the planning of significant changes that affect the software project.
 - ☐ Risk priorities and software risk management plans are reviewed and revised at these reassessment points.
 - ☐ Information obtained from monitoring the risks is used to refine the risk assessments and software risk management plans.
7. The software engineering group and other affected groups and individuals are included in the communications on the software risks, the software risk management plans, and the results of risk mitigation.

Examples of affected groups and individuals include:

- customer,
- subcontractors,
- end users,
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

Activity 11

Reviews of the software project are periodically performed to determine the actions needed to bring the software project's performance and results in line with the current and projected needs of the business, customer, and end users, as appropriate.

(Activity 11)

Examples of actions include:

- accelerating the schedule,
- changing the system requirements in response to a change in market opportunities or customer and end user needs, and
- terminating the project.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the effectiveness of the integrated software management activities.**

Examples of measurements include:

- effort expended over time to manage the software project, compared to the plan;
- frequency, causes, and magnitude of replanning effort;
- for each identified software risk, the realized adverse impact compared to the estimated loss; and
- the number and magnitude of unanticipated major adverse impacts to the software project, tracked over time.

Verifying implementation

Verification 1 **The activities for managing the software project are reviewed with senior management on a periodic basis.**

(Verification 1)

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2

The activities for managing the software project are reviewed with the project manager on both a periodic and event-driven basis.

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3

The software quality assurance group reviews and/or audits the activities and work products for managing the software project and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. The process for developing and revising the project's defined software process.
2. The process for preparing the project's software development plan and software risk management plan.
3. The processes for managing the project in accordance with the project's defined software process.
4. The processes for collecting and providing appropriate data to the organization's software process database.

- (Verification 3) 5. The process for using the organization's software process database to support the software project's planning, estimating, and tracking activities

Software Product Engineering

a key process area for Level 3: Defined

The purpose of Software Product Engineering is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.

Software Product Engineering involves performing the engineering tasks to build and maintain the software using the project's defined software process (which is described in the Integrated Software Management key process area) and appropriate methods and tools.

The software engineering tasks include analyzing the system requirements allocated to software (these system requirements are described in the Requirements Management key process area), developing the software requirements, developing the software architecture, designing the software, implementing the software in the code, integrating the software components, and testing the software to verify that it satisfies the specified requirements (i.e., the system requirements allocated to software and the software requirements).

Documentation needed to perform the software engineering tasks (e.g., software requirements document, software design document, test plan, and test procedures) is developed and reviewed to ensure that each task addresses the results of predecessor tasks and the results produced are appropriate for the subsequent tasks (including the tasks of operating and

maintaining the software). When changes are approved, affected software work products, plans, commitments, processes, and activities are revised to reflect the approved changes.

Goals

- Goal 1** The software engineering tasks are defined, integrated, and consistently performed to produce the software.
- Goal 2** Software work products are kept consistent with each other.

Commitment to perform

- Commitment 1** The project follows a written organizational policy for performing the software engineering activities.

This policy typically specifies that:

1. The software engineering tasks are performed in accordance with the project's defined software process.

Refer to Activities 1 and 2 of the Integrated Software Management key process area for practices covering the project's defined software process.

2. Appropriate methods and tools are used to build and maintain the software products.
3. The software plans, tasks, and products are traceable to the system requirements allocated to software.

(Commitment 1)

The system requirements allocated to the software are referred to as "allocated requirements" in these practices.

Refer to the Requirements Management key process area for practices covering the system requirements allocated to software.

Ability to perform

Ability 1

Adequate resources and funding are provided for performing the software engineering tasks.

1. Skilled individuals are available to perform the different software engineering tasks, including:
 - ☐ software requirements analysis,
 - ☐ software design,
 - ☐ coding,
 - ☐ testing, and
 - ☐ software maintenance.
2. Tools to support the software engineering tasks are made available.

Examples of general support tools include:

- workstations,
- database management systems,
- on-line help aids,
- graphics tools,
- interactive documentation tools, and
- word processing systems.

(Ability 1)

Examples of support tools for software requirements analysis include:

- requirements tracking tools,
- specification tools,
- prototyping tools,
- modeling tools, and
- simulation tools.

Examples of support tools for software design include:

- specification tools,
- prototyping tools,
- simulation tools, and
- program design languages.

Examples of support tools for coding include:

- editors,
- compilers,
- cross-reference generators, and
- pretty printers.

Examples of support tools for software testing include:

- test management tools,
- test generators,
- test drivers,
- test profilers,
- symbolic debuggers, and
- test coverage analyzers.

Ability 2

Members of the software engineering technical staff receive required training to perform their technical assignments.

The members of the software engineering technical staff should receive training in the application domain.

Examples of training in software requirements analysis include:

- principles of analyzing software requirements;
- the existing software requirements for any existing software to be maintained;
- skills to interview end users and application domain experts in order to establish the software requirements (i.e., requirements elicitation); and
- the use of the tools, methods, conventions, and standards selected by the project for analyzing software requirements.

Examples of training in software design include:

- design concepts;
- the existing design for any existing software to be maintained; and
- use of the tools, methods, conventions, and standards selected by the project for designing software.

Examples of training in coding include:

- the selected programming language(s);
- reviewing the existing source code for any existing code to be maintained;
- use of the tools, methods, conventions, and standards selected by the project for programming; and
- unit testing techniques.

(Ability 2)

Examples of training in software testing and other verification techniques include:

- verification methods (analysis, demonstration, and inspection as well as test);
- test planning;
- use of the tools, methods, conventions, and standards selected by the project for testing and verifying the software;
- criteria for test readiness and completion; and
- measuring test coverage.

Refer to the Training Program key process area.

Ability 3

Members of the software engineering technical staff receive orientation in related software engineering disciplines.

Examples of related software engineering disciplines include:

- software requirements analysis,
- software design,
- coding,
- testing,
- software configuration management, and
- software quality assurance.

Refer to the Training Program key process area.

Ability 4

The project manager and all software managers receive orientation in the technical aspects of the software project.

(Ability 4)

Examples of orientation include:

- software engineering methods and tools,
- the application domain,
- deliverable and nondeliverable software and associated work products, and
- guidelines on how to manage the project using the chosen methods and tools.

Refer to the Training Program key process area.

Activities performed

Activity 1

Appropriate software engineering methods and tools are integrated into the project's defined software process.

Refer to Activities 1 and 2 of the Integrated Software Management key process area for practices covering the project's defined software process.

1. The software engineering tasks are integrated according to the project's defined software process.
2. Methods and tools appropriate for use on the software project are selected.

(Activity 1)

Candidate methods and tools are selected based on their applicability to the organization's standards, the project's defined software process, the existing skill base, availability of training, contractual requirements, power, ease of use, and support services.

- ☐ The rationale for selecting a particular tool or method is documented.
- 3. Configuration management models appropriate to the software project are selected and used.

Examples of configuration management models include:

- check-out/check-in models,
- composition models,
- transaction models, and
- change set models.

- 4. The tools used to develop and maintain the software products are placed under configuration management.

Refer to the Software Configuration Management key process area.

Activity 2

The software requirements are developed, maintained, documented, and verified by systematically analyzing the allocated requirements according to the project's defined software process.

- 1. The individuals involved in developing the software requirements review the allocated requirements to ensure that issues affecting the software requirements analysis are identified and resolved.

(Activity 2)

Software requirements cover the software functions and performance, the interfaces to both hardware and software, and other system components (e.g., humans).

2. Effective methods for requirements analysis are used to identify and derive the software requirements.

Examples of methods for requirements analysis include:

- functional decomposition,
- object-oriented decomposition,
- tradeoff studies,
- simulations,
- modeling,
- prototyping, and
- scenario generation.

3. The results of the requirements analysis and the rationale for the selected alternative are documented.
4. The software requirements are analyzed to ensure they are feasible and appropriate to implement in software, clearly stated, consistent with each other, testable, and complete (when considered as a set).
- ☐ Problems with the software requirements are identified and reviewed with the group responsible for the system requirements; appropriate changes are made to the allocated requirements and to the software requirements.

Refer to the Requirements Management key process area.

5. The software requirements are documented.
6. The group responsible for system and acceptance testing of the software analyzes each software requirement to verify it can be tested.

(Activity 2)

7. The methods for verifying and validating that each software requirement is satisfied are identified and documented.

Examples of verification and validation methods include:

- demonstration,
- system testing,
- acceptance testing,
- analysis, and
- inspection.

8. The software requirements document undergoes peer review before it is considered complete.

Refer to the Peer Reviews key process area.

9. The software requirements document is reviewed and approved.

Examples of individuals who review and approve the software requirements document include:

- the project manager,
- the system engineering manager,
- the project software manager, and
- the software test manager.

10. The software requirements document is reviewed with the customer and end users, as appropriate.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

(Activity 2)

11. The software requirements document is placed under configuration management.

Refer to the Software Configuration Management key process area.

12. The software requirements are appropriately changed whenever the allocated requirements change.

Refer to the Requirements Management key process area.

Activity 3

The software design is developed, maintained, documented, and verified, according to the project's defined software process, to accommodate the software requirements and to form the framework for coding.

The software design consists of the software architecture and the detailed software design.

1. Design criteria are developed and reviewed.

Examples of design criteria include:

- verifiability,
- adherence to design standards,
- ease of construction,
- simplicity, and
- ease of planning.

2. The individuals involved in the software design review the software requirements to ensure that issues affecting the software design are identified and resolved.

(Activity 3)

3. Application standards are used where appropriate.

Examples of application standards include:

- standards for operating system interfaces,
- standards for computer-human interfaces, and
- standards for networking interfaces.

4. Effective methods are used to design the software.

Examples of software design methods include:

- prototyping,
- structural models,
- design reuse,
- object-oriented design, and
- essential systems analysis.

5. The software architecture is developed early, within the constraints of the software life cycle and technology being used.

The software architecture establishes the top-level software framework with well-defined internal and external interfaces.

6. The software architecture is reviewed to ensure that architecture issues affecting the software detailed design are identified and resolved.
7. The software detailed design is developed based on the software architecture.
8. The software design (i.e., the software architecture and detailed design) is documented.

(Activity 4)

3. The sequence in which code units are developed is based on a plan that accounts for factors such as criticality, difficulty, integration and test issues, and needs of the customer and end users, as appropriate.
4. Each code unit undergoes peer review and is unit tested before the unit is considered complete.

Refer to the Peer Reviews key process area.

5. The code is placed under configuration management.

Refer to the Software Configuration Management key process area.

6. The code is appropriately changed whenever the software requirements or software design changes.

Activity 5

Software testing is performed according to the project's defined software process.

1. Testing criteria are developed and reviewed with the customer and the end users, as appropriate.
2. Effective methods are used to test the software.
3. The adequacy of testing is determined based on:
 - ☐ the level of testing performed,

(Activity 3)

- ☐ The documentation of the software design covers the software components; the internal interfaces between software components; and the software interfaces to other software systems, to hardware, and to other system components (e.g., humans).

9. The software design document undergoes peer review before the design is considered complete.

Refer to the Peer Reviews key process area.

10. The software design document is placed under configuration management.

Refer to the Software Configuration Management key process area.

11. The software design document is appropriately changed whenever the software requirements change.

Activity 4

The software code is developed, maintained, documented, and verified, according to the project's defined software process, to implement the software requirements and software design.

1. The individuals involved in coding review the software requirements and software design to ensure that issues affecting the coding are identified and resolved.
2. Effective programming methods are used to code the software.

Examples of programming methods include:

- structured programming, and
- code reuse.

(Activity 5)

Examples of levels of testing include:

- unit testing,
- integration testing,
- system testing, and
- acceptance testing.

- ☐ the test strategy selected, and

Examples of test strategies include:

- functional (black-box),
- structural (white-box), and
- statistical.

- ☐ the test coverage to be achieved.

Examples of test coverage approaches include:

- statement coverage,
- path coverage,
- branch coverage, and
- usage profile.

4. For each level of software testing, test readiness criteria are established and used.

(Activity 5)

Examples of criteria to determine test readiness include:

- software units have successfully completed a code peer review and unit testing before they enter integration testing,
- the software has successfully completed integration testing before it enters system testing, and
- a test readiness review is held before the software enters acceptance testing.

5. Regression testing is performed, as appropriate, at each test level whenever the software being tested or its environment changes.
6. The test plan, test procedures, and test cases undergo peer review before they are considered ready for use.

Refer to the Peer Reviews key process area.

7. The test plans, test procedures, and test cases are managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) must be known, and changes must be incorporated in a controlled manner.

If a greater degree of formality than is implied by "managed and controlled" is desired, the work product can be placed under configuration management, as is described in the Software Configuration Management key process area.

8. Test plans, test procedures, and test cases are appropriately changed whenever the allocated requirements, software requirements, software design, or code being tested changes.

Activity 6

Integration testing of the software is planned and performed according to the project's defined software process.

(Activity 6)

1. The plans for integration testing are documented and based on the software development plan.
2. The integration test cases and test procedures are reviewed with the individuals responsible for the software requirements, software design, and system and acceptance testing.
3. Integration testing of the software is performed against the designated version of the software requirements document and the software design document.

Activity 7

System and acceptance testing of the software are planned and performed to demonstrate that the software satisfies its requirements.

System testing is performed to ensure the software satisfies the software requirements.

Acceptance testing is performed to demonstrate to the customer and end users that the software satisfies the allocated requirements.

1. Resources for testing the software are assigned early enough to provide for adequate test preparation.

Examples of activities required to prepare for testing include:

- preparing testing documentation,
- scheduling testing resources,
- developing test drivers, and
- developing simulators.

2. System and acceptance testing are documented in a test plan, which is reviewed with, and approved by, the customer and end users, as appropriate. The test plan covers:

- ☐ the overall testing and verification approach;

(Activity 7)

- ☐ responsibilities of the developing organization, subcontractors, customer, and end users, as appropriate;
 - ☐ test facility, test equipment, and test support requirements; and
 - ☐ acceptance criteria.
3. The test cases and test procedures are planned and prepared by a test group that is independent of the software developers.
 4. The test cases are documented and are reviewed with, and approved by, the customer and end users, as appropriate, before the testing begins.
 5. Testing of the software is performed against baselined software and the baselined documentation of the allocated requirements and the software requirements.
 6. Problems identified during testing are documented and tracked to closure.

Refer to Activity 9 of the Software Project Tracking and Oversight key process area and Activity 5 of the Software Configuration Management key process area for practices covering documenting and tracking problems.

7. Test results are documented and used as the basis for determining whether the software satisfies its requirements.
8. The test results are managed and controlled.

Activity 8

The documentation that will be used to operate and maintain the software is developed and maintained according to the project's defined software process.

1. Appropriate methods and tools are used to develop the documentation.

(Activity 8)

Examples of methods and tools include:

- word processing,
- case studies, and
- documentation reuse.

2. Documentation specialists actively participate in planning, developing, and maintaining documentation.
3. Preliminary versions of the documentation are developed and made available early in the software life cycle for the customer, end users, and software maintainers, as appropriate, to review and provide feedback.

Examples of documentation include:

- training documentation,
- on-line documentation,
- the user's manual,
- the operator's manual, and
- the maintenance manual.

4. Final versions of the documentation are verified against the software baselined for software acceptance testing.
5. The documentation undergoes peer review.

Refer to the Peer Reviews key process area.

6. The documentation is managed and controlled.
7. The final documentation is reviewed and approved by the customer, end users, and software maintainers, as appropriate.

Activity 9

Data on defects identified in peer reviews and testing are collected and analyzed according to the project's defined software process.

Examples of the kinds of data to be collected and analyzed include:

- defect description,
- defect category,
- severity of the defect,
- units containing the defect,
- units affected by the defect,
- activity where the defect was introduced,
- peer review or test cases that identified the defect,
- description of the scenario being run that identified the defect, and
- expected result and actual results that identified the defect.

Activity 10

Consistency is maintained across software work products, including the software plans, process descriptions, allocated requirements, software requirements, software design, code, test plans, and test procedures.

1. Software work products are documented, and the documentation is readily available.
2. The software requirements, design, code, and test cases are traced to the source from which they were derived and to the products of the subsequent software engineering activities.
3. The documentation tracing the allocated requirements through the software requirements, design, code, and test cases is managed and controlled.
4. As understanding of the software improves, changes to the software work products, plans, process descriptions, and activities are proposed, analyzed, and incorporated as appropriate.

(Activity 10)

- ☐ The project determines the impact of the change before the change is made.
- ☐ Where changes to the allocated requirements are needed, they are approved and incorporated before any software work products or activities are changed.
- ☐ Changes to all software products, plans, process descriptions, and activities are coordinated.
- ☐ Changes are negotiated with and communicated to the affected groups.

Examples of affected groups include:

- software engineering,
- software estimating,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

- ☐ Changes are tracked to completion.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the functionality and quality of the software products.**

Examples of measurements include:

- numbers, types, and severity of defects identified in the software products tracked cumulatively and by stage; and
- allocated requirements summarized by category (e.g., security, system configuration, performance, and reliability), and traced to the software requirements and system test cases.

Measurement 2 **Measurements are made and used to determine the status of the software product engineering activities.**

Examples of measurements include:

- status of each allocated requirement throughout the life of the project;
- problem reports by severity and length of time they are open;
- change activity for the allocated requirements;
- effort to analyze proposed changes for each proposed change and cumulative totals;
- number of changes incorporated into the software baseline by category (e.g., interface, security, system configuration, performance, and usability); and
- size and cost to implement and test incorporated changes, including initial estimate and actual size and cost.

Verifying implementation

Verification 1 **The activities for software product engineering are reviewed with senior management on a periodic basis.**

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2 **The activities for software product engineering are reviewed with the project manager on both a periodic and event-driven basis.**

(Verification 2)

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3

The software quality assurance group reviews and/or audits the activities and work products for software product engineering and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify that:

1. The software requirements are reviewed to ensure that they are:
 - ☐ complete,
 - ☐ correct,
 - ☐ consistent,
 - ☐ feasible, and
 - ☐ testable.
2. Readiness and completion criteria for each software engineering task are satisfied.
3. Software products comply with the standards and requirements specified for them.
4. Required testing is performed.
5. System and acceptance testing of the software are performed according to documented plans and procedures.
6. Tests satisfy their acceptance criteria, as documented in the software test plan.
7. Tests are satisfactorily completed and recorded.

- (Verification 3)
8. Problems and defects detected are documented, tracked, and addressed.
 9. Tracing of the allocated requirements through the software requirements, design, code, and test cases is performed.
 10. The documentation used to operate and maintain the software is verified against the software baseline and any applicable allocated requirements before the software product is released to the customer or end users.

Intergroup Coordination

a key process area for Level 3: Defined

The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

Intergroup Coordination involves the software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues. Representatives of the project's engineering groups participate in establishing the system-level requirements, objectives, and plans by working with the customer and end users, as appropriate. These requirements, objectives, and plans become the basis for all engineering activities.

The technical working interfaces and interactions between groups are planned and managed to ensure the quality and integrity of the entire system. Technical reviews and interchanges are regularly conducted with representatives of the project's engineering groups to ensure that all engineering groups are aware of the status and plans of all the groups, and that system and intergroup issues receive appropriate attention.

The software-specific practices related to these engineering tasks are described in the Requirements Management and Software Product Engineering key process areas.

Goals

- Goal 1** **The customer's requirements are agreed to by all affected groups.**
- Goal 2** **The commitments between the engineering groups are agreed to by the affected groups.**
- Goal 3** **The engineering groups identify, track, and resolve intergroup issues.**

Commitment to perform

- Commitment 1** **The project follows a written organizational policy for establishing interdisciplinary engineering teams.**

This policy typically specifies that:

1. The system requirements and project-level objectives for the project are defined and reviewed by all affected groups.

Examples of affected groups include:

- software engineering,
- software estimating,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

2. The engineering groups coordinate their plans and activities.

- (Commitment 1)** 3. Managers are responsible for establishing and maintaining an environment to facilitate interaction, coordination, support, and teamwork between the project's engineering groups, between the project and the customer or end users, as appropriate, and throughout the organization.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

Ability to perform

Ability 1 Adequate resources and funding are provided for coordinating the software engineering activities with other engineering groups.

Ability 2 The support tools used by the different engineering groups are compatible to enable effective communication and coordination.

Examples of support tools that should be compatible include:

- word processing systems,
- database systems,
- graphics tools,
- spreadsheet programs,
- problem tracking packages, and
- library management tools.

Ability 3 All managers in the organization receive required training in teamwork.

(Ability 3)

Examples of training include:

- building teams;
- managing teams;
- establishing, promoting, and facilitating teamwork; and
- group dynamics.

Refer to the Training Program key process area.

Ability 4

All task leaders in each engineering group receive orientation in the processes, methods, and standards used by the other engineering groups.

Refer to the Training Program key process area.

Ability 5

The members of the engineering groups receive orientation in working as a team.

Refer to the Training Program key process area.

Activities performed

Activity 1

The software engineering group and the other engineering groups participate with the customer and end users, as appropriate, to establish the system requirements.

(Activity 1)

Specifically, these groups:

1. Define the critical characteristics of the customer's and end users' requirements, as appropriate.
2. Negotiate critical dependencies.
3. Document the acceptance criteria for each product delivered to the customer or end user, as appropriate.

Activity 2

Representatives of the project's software engineering group work with representatives of the other engineering groups to monitor and coordinate technical activities and resolve technical issues.

1. The representatives of these groups monitor and coordinate technical activities by:
 - ☐ coordinating the specification and providing the technical review and approval of the system requirements and system design;

The system requirements and system design are typically the responsibility of the system engineering group, but representatives of the other engineering groups are expected to have significant involvement in these tasks.

The system requirements and system design include:

- the overall system requirements,
- the system configuration (i.e., hardware, software, and other system components),
- the allocation and tracing of requirements to these system components, and
- the definitions of the interfaces between these system components.

- ☐ providing the project-level technical review and analysis needed to manage and control changes to the system requirements and project-level objectives throughout the project's life cycle;

(Activity 2)

- ☐ tracking and reviewing the design and development activities for hardware, software, and other system components; and
- ☐ assessing, developing recommendations for, and tracking technical risks that involve more than one engineering group.

Refer to Activity 10 of the Integrated Software Management key process area for practices covering risk management.

2. The representatives of the groups handle technical issues by:
 - ☐ resolving project-level conflicts and clarifying system requirements and design issues;
 - ☐ developing joint recommendations to resolve problems; and
 - ☐ addressing process issues that span the engineering groups of the project.

Activity 3

A documented plan is used to communicate intergroup commitments and to coordinate and track the work performed.

This plan is:

1. The baseline for:
 - ☐ the project schedule,
 - ☐ the contractual and technical aspects of the project, and
 - ☐ the assignment of responsibilities to the engineering groups.
2. Used to coordinate activities between the different engineering groups.
3. Readily available to the members of all engineering groups.
4. Updated to incorporate all intergroup commitments and changes to these commitments.
5. Updated as the work progresses to reflect progress and plan changes at the project level, particularly when major project milestones are completed and when plans change significantly.

(Activity 3)

6. Reviewed and agreed to by all engineering groups and the project manager.

Activity 4

Critical dependencies between engineering groups are identified, negotiated, and tracked according to a documented procedure.

Refer to Activity 9 of the Integrated Software Management key process area for practices covering management of critical dependencies.

This procedure typically specifies that:

1. Each critical dependency is explicitly defined, including:
 - ☐ the item to be provided,
 - ☐ who will provide it,
 - ☐ when it will be provided, and
 - ☐ the criteria for acceptance.
2. Critical dependencies are negotiated between the software engineering group and other engineering groups in the project and organization.
3. Need dates and availability dates of critical dependency items are tied to the project schedule and the software schedule.
4. The agreement for each critical dependency is documented, reviewed, and approved by both the receiving group and the group responsible for providing the critical dependency item.
5. Critical dependencies are tracked on a regular basis and corrective actions are taken when appropriate.
 - ☐ Status and actual or projected completion are compared to the plan used to coordinate intergroup commitments.
 - ☐ Effects of late and early completions are evaluated for impacts on future activities and milestones.
 - ☐ Actual and potential problems are reported to the appropriate managers.

Intergroup Coordination

Level 3: Defined

Activity 5 Work products produced as input to other engineering groups are reviewed by representatives of the receiving groups to ensure that the work products meet their needs.

Activity 6 Intergroup issues not resolvable by the individual representatives of the project engineering groups are handled according to a documented procedure.

Examples of intergroup issues include:

- incompatible schedules,
- inadequate funding,
- technical risks,
- system-level design and requirements defects, and
- system-level problems.

Activity 7 Representatives of the project engineering groups conduct periodic technical reviews and interchanges.

In these meetings, the participants:

1. Provide visibility of the needs and desires of the customer and end users, as appropriate.
2. Monitor the technical activities of the project.
3. Ensure that the groups' interpretation and implementation of the technical requirements conform to the system requirements.
4. Review the commitments to determine whether they are being met.

Refer to the Software Project Tracking and Oversight key process area for practices covering reviews.

5. Review the technical risks and other technical issues.

(Activity 7)

Refer to Activity 10 of the Integrated Software Management key process area for practices covering risk management.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the intergroup coordination activities.**

Examples of measurements include:

- actual effort and other resources expended by the software engineering group for support to other engineering groups;
- actual effort and other resources expended by the other engineering groups in support of the software engineering group;
- actual completion of specific tasks and milestones by the software engineering group to support the activities of other engineering groups; and
- actual completion of specific tasks and milestones by the other engineering groups to support the activities of the software engineering group.

Verifying implementation

Verification 1 **The activities for intergroup coordination are reviewed with senior management on a periodic basis.**

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of the senior management oversight reviews.

Verification 2 **The activities for intergroup coordination are reviewed with the project manager on both a periodic and even-driven basis.**

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of the project management oversight reviews.

Verification 3 **The software quality assurance group reviews and/or audits the activities and work products for intergroup coordination and reports the results.**

Refer to the Software Quality Assurance key process area.

The software quality assurance responsibilities for this key process area may be subsumed into a quality assurance function that covers all the project engineering groups.

At a minimum, the reviews and/or audits verify:

1. The procedure for identifying, negotiating, and tracking critical dependencies between the project engineering groups.
2. The handling of intergroup issues.

Peer Reviews

a key process area for Level 3: Defined

The purpose of Peer Reviews is to remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of defects that might be prevented.

Peer Reviews involve a methodical examination of software work products by the producers' peers to identify defects and areas where changes are needed. The specific products that will undergo a peer review are identified in the project's defined software process and scheduled as part of the software project planning activities, as described in Integrated Software Management.

This key process area covers the practices for performing peer reviews. The practices identifying the specific software work products that undergo peer review are contained in the key process areas that describe the development and maintenance of each software work product.

Goals

- | | |
|--------|---|
| Goal 1 | Peer review activities are planned. |
| Goal 2 | Defects in the software work products are identified and removed. |

Commitment to perform

Commitment 1 **The project follows a written organizational policy for performing peer reviews.**

This policy typically specifies that:

1. The organization identifies a standard set of software work products that will undergo peer review.
2. Each project identifies the software work products that will undergo peer review.

Refer to Activity 1 of the Integrated Software Management key process area and Activity 2 of the Organization Process Definition key process area for practices covering the identification of software products that undergo peer review.

Examples of software work products include:

- operational software and support software,
 - deliverable and nondeliverable software work products,
 - software (e.g., source code) and nonsoftware work products (e.g., documents), and
 - process descriptions.

3. Peer reviews are led by trained peer review leaders.
4. Peer reviews focus on the software work product being reviewed and not on the producer.
5. Results of the peer reviews are not used by management to evaluate the performance of individuals.

Ability to perform

Ability 1

Adequate resources and funding are provided for performing peer reviews on each software work product to be reviewed.

Resources and funding are provided to:

1. Prepare and distribute the peer review materials.
2. Lead the peer review.
3. Review the materials.
4. Participate in the peer review and any follow-up reviews required based on the defects identified in the peer review.
5. Monitor the rework of the software work product based on the defects identified in the peer review.
6. Collect and report the data resulting from the peer reviews.

Ability 2

Peer review leaders receive required training in how to lead peer reviews.

(Ability 2)

Examples of training include:

- the objectives, principles, and methods of peer reviews;
- planning and organizing a peer review;
- evaluating readiness and completion criteria for a peer review;
- conducting and facilitating a peer review;
- reporting the results of a peer review;
- tracking and confirming rework to address the actions identified in a peer review; and
- collecting and reporting the data required for the peer reviews.

Refer to the Training Program key process area.

Ability 3

Reviewers who participate in peer reviews receive required training in the objectives, principles, and methods of peer reviews.

Examples of training include:

- types of peer reviews (e.g., reviews of software requirements, software design, code, and software test procedures);
- the objectives, principles, and methods of peer reviews;
- roles of reviewers; and
- estimating the effort for preparing and participating in peer reviews.

Refer to the Training Program key process area.

Activities performed

Activity 1

Peer reviews are planned, and the plans are documented.

These plans:

1. Identify the software work products that will undergo peer review.
 - ☐ The software work products selected include the set identified in the organization's standard software process.

Refer to Activity 2 of the Organization Process Definition key process area for practices covering the organization's standard software process.

2. Specify the schedule of peer reviews.

For peer reviews that are scheduled to occur in the near future, the trained peer review leaders and the other reviewers for each peer review are identified.

Activity 2

Peer reviews are performed according to a documented procedure.

This procedure typically specifies that:

1. Peer reviews are planned and led by trained peer review leaders.
2. Review materials are distributed to the reviewers in advance so they can adequately prepare for the peer review.

(Activity 2)

The review materials should include the relevant inputs to the development of the software work product undergoing peer review.

Examples of relevant input include:

- the objectives of the software work product,
- the applicable standards,
- the relevant requirements for a design module, or
- the relevant detailed design for a code module.

3. Reviewers have assigned roles in peer reviews.
4. Readiness and completion criteria for the peer reviews are specified and enforced.
 - ☐ Issues in satisfying these criteria are reported to the appropriate managers.
5. Checklists are used to identify criteria for the review of the software work products in a consistent manner.
 - ☐ The checklists are tailored to the specific type of work product and peer review.

Examples of items addressed by tailoring the checklist include:

- compliance with standards and procedures,
- completeness,
- correctness,
- rules of construction, and
- maintainability.

- ☐ The checklists are reviewed by the checklist developers' peers and potential users.
6. Actions identified in the peer reviews are tracked until they are resolved.

(Activity 2)

7. The successful completion of peer reviews, including the rework to address the items identified in the peer reviews, is used as a completion criterion for the associated task.

Activity 3

Data on the conduct and results of the peer reviews are recorded.

Examples of data include:

- identification of the software work product reviewed,
- size of the software work product,
- size and composition of the review team,
- preparation time per reviewer,
- length of the review meeting,
- types and number of defects found and fixed, and
- rework effort.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the peer review activities.

Examples of measurements include:

- number of peer reviews performed compared to the plan,
- overall effort expended on peer reviews compared to the plan, and
- number of work products reviewed compared to the plan.

Verifying implementation

Verification 1

The software quality assurance group reviews and/or audits the activities and work products for peer reviews and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify that:

1. The planned peer reviews are conducted.
2. The peer review leaders are adequately trained for their roles.
3. The reviewers are properly trained or experienced in their roles.
4. The process for preparing for the peer reviews, conducting the peer reviews, and performing the follow-up actions are followed.
5. Reporting of peer review data is complete, accurate, and timely.

Capability Maturity Model

Level 4: Managed

Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures.

Key process areas

<u>Quantitative Process Management</u>	<u>L4-1</u>
<u>Software Quality Management</u>	<u>L4-19</u>

Quantitative Process Management

a key process area for Level 4: Managed

The purpose of Quantitative Process Management is to control the process performance of the software project quantitatively. Software process performance represents the actual results achieved from following a software process.

Quantitative Process Management involves establishing goals for the performance of the project's defined software process, which is described in the Integrated Software Management key process area, taking measurements of the process performance, analyzing these measurements, and making adjustments to maintain process performance within acceptable limits. When the process performance is stabilized within acceptable limits, the project's defined software process, the associated measurements, and the acceptable limits for the measurements are established as a baseline and used to control process performance quantitatively.

The organization collects process performance data from the software projects and uses these data to characterize the process capability (i.e., the process performance a new project can expect to attain) of the organization's standard software process, which is described in the Organization Process Definition key process area. Process capability describes the range of expected results from following a software process (i.e., the most likely outcomes that are expected from the next software project the organization undertakes). These process capability data are, in

turn, used by the software projects to establish and revise their process performance goals and to analyze the performance of the projects' defined software processes.

Goals

- | | |
|---------------|---|
| Goal 1 | The quantitative process management activities are planned. |
| Goal 2 | The process performance of the project's defined software process is controlled quantitatively. |
| Goal 3 | The process capability of the organization's standard software process is known in quantitative terms. |

Commitment to perform

- | | |
|---------------------|--|
| Commitment 1 | The project follows a written organizational policy for measuring and quantitatively controlling the performance of the project's defined software process. |
|---------------------|--|

This policy typically specifies that:

1. Each project implements a documented plan to bring the project's defined software process under quantitative control.

(Commitment 1)

The term "quantitative control" implies any quantitative or statistically-based technique appropriate to analyze a software process, identify special causes of variations in the performance of the software process, and bring the performance of the software process within well-defined limits.

A special cause of variation is some transient circumstance (such as a specific local condition, a single machine, a single individual, or a small group of people performing in an unexpected way) that causes an unexpected, transient change in the process performance.

2. Sensitive data relating to individuals' performance are protected, and access to these data is appropriately controlled.

Use of measurement data to evaluate individuals will negatively affect the correctness and usefulness of the measurement data that are reported.

Commitment 2 The organization follows a written policy for analyzing the process capability of the organization's standard software process.

This policy typically specifies that:

1. The projects' measurements of process performance are analyzed to establish and maintain a process capability baseline for the organization's standard software process.

The process capability baseline includes:

- the description of the organization's standard software process,
- the standard definitions of the measurements, and
- the expected range of values for the measurements.

- (Commitment 2) 2. The process capability baseline for the organization's standard software process is used by the software projects in establishing their process performance goals.

Ability to perform

Ability 1

A group that is responsible for coordinating the quantitative process management activities for the organization exists.

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time. Considerations when implementing a group include assigned tasks or activities, the size of the project, the organizational structure, and the organizational culture. Some groups, such as the software quality assurance group, are focused on project activities, and others, such as the software engineering process group, are focused on organization-wide activities.

1. This group is either part of the group responsible for the organization's software process activities (e.g., software engineering process group) or its activities are closely coordinated with that group.

Ability 2

Adequate resources and funding are provided for the quantitative process management activities.

1. The managers and task leaders of the software engineering groups and other software-related groups perform the project's quantitative process management activities.

(Ability 2)

Examples of software-related groups include:

- software quality assurance,
- software configuration management, and
- documentation support.

2. An organization-wide measurement program exists.

The organization's measurement program includes:

- the definition of the organization-wide measurements,
- the collection of the organization's measurement data,
- the analysis of the organization's measurement data, and
- the quantitative measurement goals for the organization.

3. Tools to support quantitative process management are made available.

Examples of support tools include:

- software source code analyzers,
- automated test coverage analyzers,
- database systems,
- quantitative analysis packages, and
- problem tracking packages.

Ability 3

Support exists for collecting, recording, and analyzing data for selected process and product measurements.

The product data referred to in these practices are product measurements used in analyzing the software process.

Ability 4

The individuals implementing or supporting quantitative process management receive required training to perform these activities.

Examples of training include:

- modeling and analyzing the software process;
- selecting, collecting, and validating process measurement data; and
- applying basic quantitative methods and analysis techniques (e.g., estimation models, Pareto diagrams, and control charts).

Refer to the Training Program key process area.

Ability 5

The members of the software engineering group and other software-related groups receive orientation on the goals and value of quantitative process management.

Refer to the Training Program key process area.

Activities performed

Activity 1

The software project's plan for quantitative process management is developed according to a documented procedure.

This procedure typically specifies that:

1. The quantitative process management plan is based on:
 - ☐ the organization's strategic goals for product quality, productivity, and product development cycle time;
 - ☐ the organization's measurement program;

(Activity 1)

- ☐ the organization's standard software process;
- ☐ the project's goals for the software product's quality, productivity, and product development cycle time;
- ☐ the measured performance of other projects' defined software processes; and
- ☐ the description of the project's defined software process.

2. The plan undergoes peer review.

Refer to the Peer Reviews key process area.

3. The plan is reviewed by the group responsible for the organization's software process activities (e.g., the software engineering process group).

4. The plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of formality than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 2

The software project's quantitative process management activities are performed in accordance with the project's quantitative process management plan.

The plan covers:

1. The goals and objectives of the quantitative process management activities.

(Activity 2)

2. The software tasks or other software activities that will be measured and analyzed.
3. The instrumentation of the project's defined software process.

The instrumentation is based on the organization's measurement program, the description of the organization's standard software process, and the description of the project's defined software process.

4. The quantitative process management activities to be performed and the schedule for these activities.

In addition to current organizational and project needs, measurements that may be useful to future efforts are included.

5. The groups and individuals responsible for the quantitative process management activities.
6. The resources required to perform the quantitative process management activities, including staff and tools.
7. The procedures to be followed in performing the quantitative process management activities.

Activity 3

The strategy for the data collection and the quantitative analyses to be performed are determined based on the project's defined software process.

The attributes of the project's defined software process that are considered include:

1. The tasks, the activities, and their relationships to each other.

(Activity 3)

2. The software work products and their relationships to each other and to the project's defined software process.
3. The process control points and data collection points.

Activity 4

The measurement data used to control the project's defined software process quantitatively are collected according to a documented procedure.

This procedure typically specifies that:

1. The measurement data collected support the organization's and the software project's measurement goals and objectives.
2. The specific measurement data to be collected, their precise definitions, the intended use and analysis of each measurement, and the process control points at which they will be collected are defined.

Examples of measurement data include:

- estimated/planned versus actual data on software size, cost, and schedule;
- productivity data;
- quality measurements as defined in the software quality plan;
- coverage and efficiency of peer reviews;
- effectiveness of training;
- test coverage and efficiency;
- software reliability measures;
- number and severity of defects found in the software requirements;
- number and severity of defects found in the software code; and
- number and rate of closure on action items.

3. The measurements are chosen from the entire software life cycle (e.g., both the development and post-development stages).

(Activity 4)

4. The measurements cover the properties of the key software process activities and major software work products.
5. The measurement data that relate to the organization's standard software process are uniformly collected across the software projects.
6. The measurements to be controlled are a natural result of the software activities where possible.
7. The measurements are selected to support predefined analysis activities.

In some cases, measurements may be research oriented and should be explicitly designated as such.

8. The validity of the measurement data is independently assessed.
9. The collected measurement data are stored in the organization's software process database as appropriate.

Refer to Activity 5 of the Organization Process Definition key process area for practices covering the organization's software process database.

Activity 5

The project's defined software process is analyzed and brought under quantitative control according to a documented procedure.

This procedure typically specifies that:

1. The specific data analysis activities are predefined.

(Activity 5)

The description of the data analysis activities covers:

- input data required,
- tools used,
- data manipulations performed,
- information to be derived, and
- decision criteria used in performing the analysis and deciding what actions to take as a result of the analysis.

Examples of analysis techniques include:

- Pareto diagrams,
- control charts,
- trend diagrams, and
- scatter diagrams.

2. Measurement data on the process activities throughout the project's defined software process are identified, collected, and analyzed.
3. The selected measurements appropriately characterize the process they represent.
4. The expected values for mean and variance are specified for each measurement.
5. The acceptable limits for each measurement are defined and the project's process performance baseline is established.

An example of establishing acceptable limits is to calculate the historical deviation from the mean performance of the process.

6. The actual values of each measurement are compared to the expected values of the mean and variance.

(Activity 5)

Examples of comparing actual process performance to defined acceptable limits include:

- comparing the peer review hours spent per thousand lines of source code to upper and lower limits determined by analyzing historical data; and
- comparing the expansion ratio of software requirements (e.g., number of "shalls") into the number of lines of source code to upper and lower limits determined by analyzing historical data.

7. Adjustments are made to bring the actual process performance in line with the defined acceptable limits, as appropriate.
8. When the project's defined software process is controlled quantitatively, baselines are established for:
 - ☐ the definition of the measurements,
 - ☐ the actual measurement data, and
 - ☐ the acceptable limits for the measurements.
9. The process performance baseline for the software project is managed and controlled.

Activity 6

Reports documenting the results of the software project's quantitative process management activities are prepared and distributed.

1. The results of the data analysis are reviewed with those affected by the data before they are reported to anyone else.
2. The software managers, software task leaders, and senior management receive regular reports appropriate for their needs.
3. The software quality assurance group receives regular reports appropriate to its needs.
4. The project manager, senior managers, software managers, and software task leaders receive specialized reports on request.

Activity 7

The process capability baseline for the organization's standard software process is established and maintained according to a documented procedure.

This procedure typically specifies that:

1. The project's software process data, as summarized in its process performance baseline, are recorded in the organization's software process database.

Refer to Activity 5 of the Organization Process Definition key process area for practices covering the organization's software process database.

2. The process performance baseline for each project's defined software process is incorporated, as appropriate, into the process capability baseline for the organization's standard software process.
3. The process capability baseline for the organization's standard software process is documented.
4. Process capability trends for the organization's standard software process are examined to predict likely problems or opportunities for improvements.

Examples of using capability trends include:

- predicting the occurrence of software defects and comparing the predictions to actuals, and
- predicting the distribution and characteristics of defects remaining in a product based on the data from peer reviews and/or test.

(Activity 7)

Examples of areas that are likely sources of defects include:

- items for estimating and planning,
- activities performed early in the software life cycle such as requirements analysis,
- major documentation items,
- items and activities that have been prone to defect insertion in the past,
- activities for implementing changes and fixing defects, and
- labor-intensive activities.

Examples of areas that are likely opportunities for improvement include:

- activities that other projects and organizations have successfully automated,
- nondeliverable and support items and activities such as training and tools,
- quality oriented activities such as peer reviews and testing, and
- labor intensive activities.

5. The process capability baseline for the organization's standard software process is managed and controlled.
6. When a software project that is substantially different from past projects is undertaken, a new process performance baseline is established for that project as part of tailoring the organization's standard software process.

Refer to Activity 1 of the Integrated Software Management key process area for practices covering the project's tailoring of the organization's standard software process.

(Activity 7)

Examples of substantial differences include:

- new application domains,
- use of radically different technologies, and
- significant change in the size of the application.

7. Changes to the organization's standard software process are tracked and analyzed to assess their effects on the process capability baseline.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the activities for quantitative process management.**

Examples of measurements include:

- the cost over time for the quantitative process management activities, compared to the plan; and
- the accomplishment of schedule milestones for quantitative process management activities, compared to the approved plan (e.g., establishing the process measurements to be used on the project, determining how the process data will be collected, and collecting the process data).

Verifying implementation

Verification 1 **The activities for quantitative process management are reviewed with senior management on a periodic basis.**

(Verification 1)

Refer to Verification 1 of the Organization Process Focus key process area and Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2

The software project's activities for quantitative process management are reviewed with the project manager on both a periodic and event-driven basis.

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3

The software quality assurance group reviews and/or audits the activities and work products for quantitative process management and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify that:

1. The plans for the quantitative process management activities are followed.
2. The procedures for quantitative process management are followed.
3. The collection and analysis of quantitative process management data are performed as required, including verification that:
 - ☐ the needed data exist,
 - ☐ the needed data are collected,
 - ☐ the data collected are needed,

(Verification 3)

- ☐ the data collected support the goals and objectives of the organization's measurement program,
- ☐ the cost of collecting the data is justified by the usefulness of the data,
- ☐ the data are collected at the correct point in the software life cycle,
- ☐ the data are accurate and correct,
- ☐ the data are timely, and
- ☐ the confidentiality of the data is properly protected.

Software Quality Management

a key process area for Level 4: Managed

The purpose of Software Quality Management is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

Software Quality Management involves defining quality goals for the software products, establishing plans to achieve these goals, and monitoring and adjusting the software plans, software work products, activities, and quality goals to satisfy the needs and desires of the customer and end user for high quality products.

The practices of Software Quality Management build on the practices of the Integrated Software Management and Software Product Engineering key process areas, which establish and implement the project's defined software process, and the Quantitative Process Management key process area, which establishes a quantitative understanding of the ability of the project's defined software process to achieve the desired results.

Quantitative goals are established for the software products based on the needs of the organization, the customer, and the end users. So that these goals may be achieved, the organization establishes strategies and plans, and the project specifically adjusts its defined software process, to accomplish the quality goals.

Goals

- Goal 1** **The project's software quality management activities are planned.**
- Goal 2** **Measurable goals for software product quality and their priorities are defined.**
- Goal 3** **Actual progress toward achieving the quality goals for the software products is quantified and managed.**

Commitment to perform

- Commitment 1** **The project follows a written organizational policy for managing software quality.**

This policy typically specifies that:

1. The project's software quality management activities support the organization's commitment to improve the quality of the software products.

Improvements to the process that increase software product quality are a top priority of the organization.

Each new software product release should be measurably better than its predecessor or leading competitor.

2. The project defines and collects the measurements used for software quality management based on the project's defined software process.
3. The project defines the quality goals for the software products and monitors its progress towards them.

- (Commitment 1)** 4. Responsibilities for software quality management are defined and assigned to the software engineering group and other software-related groups.

Examples of software-related groups include:

- software quality assurance,
- software configuration management, and
- documentation support.

- ☐ Criteria are established to enable the groups to determine their success in achieving the quality goals for the software products.

Ability to perform

Ability 1

Adequate resources and funding are provided for managing the quality of the software products.

1. Specialty engineers in areas such as safety and reliability are available to help set the software quality goals and review progress towards the goals.
2. Tools to support predicting, measuring, tracking, and analyzing software quality are made available.

Examples of support tools include:

- data collection tools,
- database systems,
- spreadsheet programs,
- software life-cycle simulators,
- quantitative analysis tools, and
- code audit tools.

Ability 2

The individuals implementing and supporting software quality management receive required training to perform their activities.

Examples of training include:

- planning quality commitments and goals for the product,
- measuring product and process quality, and
- controlling product quality using the defined software process.

Refer to the Training Program key process area.

Ability 3

The members of the software engineering group and other software-related groups receive required training in software quality management.

Examples of training include:

- understanding the goals and benefits of quantitatively managing product quality,
- collecting measurement data,
- understanding the quality measurements for the software process and product, and
- planning and controlling the quality of the software product.

Refer to the Training Program key process area.

Activities performed

Activity 1

The project's software quality plan is developed and maintained according to a documented procedure.

This procedure typically specifies that:

1. An understanding of the software quality needs of the organization, customer, and end users is developed as appropriate.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

Examples of ways to measure the customer's and end users' software quality needs include:

- surveys,
- focus groups, and
- product evaluations by users.

2. The software quality needs and priorities of the organization, customer, and end user are traceable to the system requirements allocated to software and the software quality goals.

An example of a method to trace these needs and priorities is Quality Function Deployment (QFD).

An example of tracing needs and priorities to the software quality goals for the product is establishing targets for the number of post-delivery defects and performing predictive exercises as the product matures to assess the likelihood of meeting those goals.

(Activity 1)

The system requirements allocated to the software are referred to as "allocated requirements" in these practices.

Refer to the Requirements Management key process area for practices covering the system requirements allocated to software.

3. The capability of the project's defined software process to satisfy the software quality goals is assessed and documented.

Techniques such as Quality Function Deployment and Taguchi's method for robust design can be used to relate the quality goals of a product to the process capability.

4. The software quality plan satisfies the quality plans of the organization, as appropriate.
5. The software quality plan is based on plans for previous or current projects in the organization, as appropriate.
6. The software quality plan is updated at the start of the project, at major project milestones, and whenever the allocated requirements change significantly.
7. The software quality plan undergoes peer review.

Refer to the Peer Reviews key process area.

8. The software quality plan is reviewed by affected groups and individuals.

(Activity 1)

Examples of affected groups and individuals include:

- the customer,
- the end user,
- software engineering (including all subgroups, such as software design),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

9. Senior management reviews the software quality plans.
10. The software quality plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of formality than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

11. The software quality plan is available to all affected groups and individuals.

Activity 2

The project's software quality plan is the basis for the project's activities for software quality management.

(Activity 2)

The plan covers:

1. The points in the process where software quality is measured.
2. The high-leverage quality goals for the software products.

High-leverage quality goals for the software products are those that provide the greatest customer satisfaction at the least cost, or the "must haves" from the customer or end user.

3. The actions that the software project will implement to improve on past quality performance.
4. The activities to measure software product quality.

Examples of software activities to measure software product quality include:

- peer reviews,
- prototype development,
- product simulation, and
- testing.

5. Quality goals for software work products, as appropriate.

Examples of quality goals for software products that are appropriate to document in the project's software quality plan include:

- the characteristics that are planned to be met; and
- the critical characteristics that, if not met, would make the product undesirable or not needed by the customers or end users.

6. The actions that will be taken when the software product quality is projected not to meet the quality goals.

Activity 3

The project's quantitative quality goals for the software products are defined, monitored, and revised throughout the software life cycle.

1. Characteristics of product quality that describe how well the software product will perform or how well it can be developed and maintained are identified.

Examples of software product quality characteristics include:

- functionality,
- reliability,
- maintainability, and
- usability.

2. The measurements used to quantify the characteristics of software product quality are identified.

Examples of activities to identify measurements for software product quality include:

- reviewing prior performance data and customer requirements,
- developing prototypes,
- expressing intermediate software products in formal representations,
- using formal software engineering methods, and
- conducting tests.

3. For each characteristic of software product quality, measurable, numeric values, based on the required and desired values, are selected as quality goals for the product.

(Activity 3)

Examples of possible quality goals for the software product's reliability include:

- the mean time between failure as specified in the requirements,
- the mean time between failure that must be achieved (as determined by analysis and experimentation), and
- the mean time between failure that is planned to be achieved.

4. Quality goals for the software products are documented in the project's software quality plan.

Examples of quality goals for software products that are appropriate to document in the project's software quality plan include:

- the characteristics that are planned to be met; and
- the critical characteristics that, if not met, would make the product undesirable or not needed by the customers or end users.

5. Quality goals for each software life-cycle stage are defined and documented.

Examples of software life-cycle stages include:

- software requirements,
- software design,
- coding, and
- software test.

(Activity 3)

Examples of quality goals related to software life-cycle stages include:

- product defects related to each software life-cycle stage will be reduced from the previous product release by some predetermined percentage, and
- a predetermined percentage of predicted defects will be found by the end of the test cycle.

6. Quality goals for the software products and software life-cycle stages are revised as understanding of the products and understanding of the organization's, customer's, and end users' needs evolve.

Activity 4

The quality of the project's software products is measured, analyzed, and compared to the products' quantitative quality goals on an event-driven basis.

Refer to the Quantitative Process Management key process area for practices covering use of measurement data.

1. The software tasks are planned and performed to address the project's software quality goals. At the beginning of a software task, the team performing the task:
- ☐ reviews the quality goals for the software product,
 - ☐ determines the quality goals applicable to the software task,
 - ☐ identifies its plans to achieve the software quality goals, and
 - ☐ reviews changes made to the process to meet the software quality goals.

An example of a change is revising a peer review checklist to address defects that have been found to escape peer reviews.

2. The quality of the software work products of each software life-cycle stage are measured.

(Activity 4)

Examples of methods to measure the quality of work products include:

- peer reviews,
- simulation, and
- testing.

3. The quality measurements are analyzed and compared to the software quality goals to determine whether the quality goals are satisfied.
4. Appropriate actions, consistent with the software quality plan, are taken to bring the quality measures of the products in line with the software quality goals.
5. When it is determined that the software quality goals conflict (that is, one goal cannot be achieved without compromising another goal), actions are taken to resolve the conflict.
 - ☐ The cost for achieving the software quality goals is analyzed.
 - ☐ Alternative software quality goals are considered in light of long-term business strategies as well as short-term priorities
 - ☐ The customer and end users participate in quality tradeoff decisions, as appropriate.
 - ☐ The software work products and plans are revised, as appropriate, to reflect the results of the tradeoffs.

Activity 5

The software project's quantitative quality goals for the products are allocated appropriately to the subcontractors delivering software products to the project.

Refer to Activity 1 of the Software Subcontract Management key process area.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the software quality management activities.**

Examples of measurements include:

- the cost of poor quality (based on the known quality measurements to whatever degree of accuracy they can be collected), and
- the costs for achieving the quality goals.

Verifying implementation

Verification 1 **The activities for software quality management are reviewed with senior management on a periodic basis.**

Refer to Verification 1 of the Software Project Tracking and Oversight key process area for practices covering the typical content of senior management oversight reviews.

Verification 2 **The activities for software quality management are reviewed with the project manager on both a periodic and event-driven basis.**

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3 **The software quality assurance group reviews and/or audits the activities and work products for software quality management and reports the results.**

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. The preparation of the project's software quality plan.
2. The process for establishing and tracking the software quality goals.

Capability Maturity Model

Level 5: Optimizing

Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies.

Key process areas

Defect Prevention	L5-1
Technology Change Management	L5-17
Process Change Management	L5-31

Defect Prevention

a key process area for Level 5: Optimizing

The purpose of Defect Prevention is to identify the cause of defects and prevent them from recurring.

Defect Prevention involves analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future. The defects may have been identified on other projects as well as in earlier stages or tasks of the current project. Defect prevention activities are also one mechanism for spreading lessons learned between projects.

Trends are analyzed to track the types of defects that have been encountered and to identify defects that are likely to recur. Based on an understanding of the project's defined software process and how it is implemented (as described in the Integrated Software Management and Software Product Engineering key process areas), the root causes of the defects and the implications of the defects for future activities are determined.

Both the project and the organization take specific actions to prevent recurrence of the defects. Some of the organizational actions may be handled as described in the Process Change Management key process area.

Goals

- Goal 1** **Defect prevention activities are planned.**
- Goal 2** **Common causes of defects are sought out and identified.**
- Goal 3** **Common causes of defects are prioritized and systematically eliminated.**

Commitment to perform

- Commitment 1** **The organization follows a written policy for defect prevention activities.**

This policy typically specifies that:

1. Long-term plans and commitments are established for funding, staffing, and other resources for defect prevention.
2. The resources needed are allocated for the defect prevention activities.
3. Defect prevention activities are implemented across the organization to improve the software processes and products.
4. The results of the defect prevention activities are reviewed to ensure the effectiveness of those activities.
5. Management and technical actions identified as a result of the defect prevention activities are addressed.

- Commitment 2** **The project follows a written organizational policy for defect prevention activities.**

(Commitment 2) This policy typically specifies that:

1. Defect prevention activities are included in each project's software development plan.
2. The resources needed are allocated for the defect prevention activities.
3. Project management and technical actions identified as a result of the defect prevention activities are addressed.

Ability to perform

Ability 1

An organization-level team to coordinate defect prevention activities exists.

1. This team is either part of the group responsible for the organization's software process activities (e.g., software engineering process group) or its activities are closely coordinated with that group.

Refer to the Organization Process Focus key process area.

Ability 2

A team to coordinate defect prevention activities for the software project exists.

1. This team is closely tied to the team responsible for developing and maintaining the project's defined software process.

Members of the team coordinating defect prevention activities are usually assigned to this team on a part-time basis and have other software engineering activities as their primary responsibility.

(Ability 2)

Refer to Activities 1 and 2 of the Integrated Software Management key process area for practices covering developing and maintaining the project's defined software process.

Ability 3

Adequate resources and funding are provided for defect prevention activities at the project and organization levels.

1. Defect prevention activities are planned into each person's responsibilities, as appropriate.

Examples of defect prevention activities include:

- task kick-off meetings,
- causal analysis meetings,
- reviewing and planning proposed actions, and
- implementing actions.

2. Management participation in the defect prevention activities is planned.
3. Each software project is represented on the team coordinating defect prevention activities for the organization, as appropriate.
4. Tools to support defect prevention activities are made available.

Examples of support tools include:

- statistical analysis tools, and
- database systems.

Ability 4

Members of the software engineering group and other software-related groups receive required training to perform their defect prevention activities.

(Ability 4)

Examples of software-related groups include:

- software quality assurance,
- software configuration management, and
- documentation support.

Examples of training include:

- defect prevention methods,
- conduct of task kick-off meetings,
- conduct of causal analysis meetings, and
- statistical methods (e.g., cause/effect diagrams and Pareto analysis).

Refer to the Training Program key process area.

Activities performed

Activity 1

The software project develops and maintains a plan for its defect prevention activities.

This plan:

1. Identifies the defect prevention activities (e.g., task kick-off and causal analysis meetings) that will be held.
2. Specifies the schedule of defect prevention activities.
3. Covers the assigned responsibilities and resources required, including staff and tools.
4. Undergoes peer review.

(Activity 1)

Refer to the Peer Reviews key process area.

Activity 2

At the beginning of a software task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities.

Kick-off meetings are held to familiarize the members of the team with the details of the implementation of the process, as well as any recent changes to the process.

These kick-off meetings cover:

1. The software process, standards, procedures, methods, and tools applicable to the task, with an emphasis on recent changes.

Changes may be implemented as an experiment to evaluate a recommendation from a previous causal analysis meeting.

2. The inputs required and available for the task.
3. The outputs to be produced with examples, if available.
4. The methods to be used to evaluate the outputs.
5. The methods to be used to verify adherence to the software process.
6. A list of errors that are commonly made or introduced during the current stage and recommended preventive actions for these errors.
7. The team assignments.
8. The task schedule.

(Activity 2)

9. The software product quality goals for the task and software project.

Refer to the Software Quality Management key process area.

Activity 3

Causal analysis meetings are conducted according to a documented procedure.

This procedure typically specifies that:

1. Each team that performs a software task conducts causal analysis meetings.
 - ☐ A causal analysis meeting is conducted shortly after the task is completed.
 - ☐ Meetings are conducted during the software task if and when the number of defects uncovered warrants the additional meetings.
 - ☐ Periodic causal analysis meetings are conducted after software products are released to the customer, as appropriate.
 - ☐ For software tasks of long duration, periodic in-process defect prevention meetings are conducted, as appropriate.

An example of a long duration task is a level-of-effort, customer support task.

2. The meetings are led by a person trained in conducting causal analysis meetings.
3. Defects are identified and analyzed to determine their root causes.

An example of a method to determine root causes is cause/effect diagrams.

4. The defects are assigned to categories of root causes.

(Activity 3)

Examples of defect root cause categories include:

- inadequate training,
- breakdown of communications,
- not accounting for all details of the problem, and
- making mistakes in manual procedures (e.g., typing).

5. Proposed actions to prevent the future occurrence of identified defects and similar defects are developed and documented.

Examples of proposed actions include modifications to:

- the process,
- training,
- tools,
- methods,
- communications, and
- software work products.

6. Common causes of defects are identified and documented.

Examples of common causes include:

- frequent errors made in invoking a certain system function, and
- frequent errors made in a related group of software units.

7. The results of the meeting are recorded for use by the organization and other projects.

Activity 4

Each of the teams assigned to coordinate defect prevention activities meets on a periodic basis to review and coordinate implementation of action proposals from the causal analysis meetings.

(Activity 4)

The teams involved may be at the organization or project level.

The teams:

1. Review the output from the causal analysis meetings and select action proposals that will be addressed.
2. Review action proposals that have been assigned to them by other teams coordinating defect prevention activities in the organization and select action proposals that will be addressed.
3. Review actions taken by the other teams in the organization to assess whether these actions can be applied to their activities and processes.
4. Perform a preliminary analysis of the action proposals and set their priorities.

Priority is usually nonrigorous and is based on an understanding of:

- the causes of defects,
- the implications of not addressing the defects,
- the cost to implement process improvements to prevent the defects, and
- the expected impact on software quality.

An example of a technique used to set priorities for the action proposals is Pareto analysis.

5. Reassign action proposals to teams at another level in the organization, as appropriate.
6. Document their rationale for decisions and provide the decision and the rationale to the submitters of the action proposals.
7. Assign responsibility for implementing the action items resulting from the action proposals.

(Activity 4)

- ☐ Implementation of the action items includes making immediate changes to the activities that are within the purview of the team and arranging for other changes.
 - ☐ Members of the team usually implement the action items, but, in some cases, the team can arrange for someone else to implement an action item.
8. Review results of defect prevention experiments and take actions to incorporate the results of successful experiments into the rest of the project or organization, as appropriate.

Examples of defect prevention experiments include:

- using a temporarily modified process, and
- using a new tool.

9. Track the status of the action proposals and action items.
10. Document software process improvement proposals for the organization's standard software process and the projects' defined software processes as appropriate.

The submitters of the action proposal are designated as the submitters of the software process improvement proposals.

Refer to Activity 5 of the Process Change Management key process area for practices covering handling of software process improvement proposals.

11. Review and verify completed action items before they are closed.
12. Ensure that significant efforts and successes in preventing defects are recognized.

Activity 5

Defect prevention data are documented and tracked across the teams coordinating defect prevention activities.

1. Action proposals identified in causal analysis meetings are documented.

Examples of data that are in the description of an action proposal include:

- originator of the action proposal,
- description of the defect,
- description of the defect cause,
- defect cause category,
- stage when the defect was injected,
- stage when the defect was identified,
- description of the action proposal, and
- action proposal category.

2. Action items resulting from action proposals are documented.

Examples of data that are in the description of an action item include:

- the person responsible for implementing it,
- a description of the areas affected by it,
- the individuals who are to be kept informed of its status,
- the next date its status will be reviewed,
- the rationale for key decisions,
- a description of implementation actions,
- the time and cost for identifying the defect and correcting it, and
- the estimated cost of not fixing the defect.

3. The defect prevention data are managed and controlled.

(Activity 5)

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 6

Revisions to the organization's standard software process resulting from defect prevention actions are incorporated according to a documented procedure.

Refer to Activity 1 of the Organization Process Definition key process area for practices covering the organization's standard software process.

Activity 7

Revisions to the project's defined software process resulting from defect prevention actions are incorporated according to a documented procedure.

Refer to Activity 2 of the Integrated Software Management key process area for practices covering the project's defined software process.

Activity 8

Members of the software engineering group and software-related groups receive feedback on the status and results of the organization's and project's defect prevention activities on a periodic basis.

(Activity 8)

The feedback provides:

1. A summary of the major defect categories.
2. The frequency distribution of defects in the major defect categories.
3. Significant innovations and actions taken to address the major defect categories.
4. A summary status of the action proposals and action items.

Examples of means to provide this feedback include:

- electronic bulletin boards,
- newsletters, and
- information flow meetings.

Measurement and analysis

Measurement 1 Measurements are made and used to determine the status of the defect prevention activities.

(Measurement 1)

Examples of measurements include:

- the costs of defect prevention activities (e.g., holding causal analysis meetings and implementing action items), cumulatively;
- the time and cost for identifying the defects and correcting them, compared to the estimated cost of not correcting the defects;
- profiles measuring the number of action items proposed, open, and completed;
- the number of defects injected in each stage, cumulatively, and over releases of similar products; and
- the number of defects.

Verifying implementation

Verification 1

The organization's activities for defect prevention are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

These reviews cover:

1. A summary of the major defect categories and the frequency distribution of defects in these categories.
2. A summary of the major action categories and the frequency distribution of actions in these categories.
3. Significant actions taken to address the major defect categories.

- (Verification 1)**
4. A summary status of the proposed, open, and completed action items.
 5. A summary of the effectiveness of and savings attributable to the defect prevention activities.
 6. The actual cost of completed defect prevention activities and the projected cost of planned defect prevention activities.

Verification 2 **The software project's activities for defect prevention are reviewed with the project manager on both a periodic and event-driven basis.**

Refer to Verification 2 of the Software Project Tracking and Oversight key process area for practices covering the typical content of project management oversight reviews.

Verification 3 **The software quality assurance group reviews and/or audits the activities and work products for defect prevention and reports the results.**

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify that:

1. The software engineering managers and technical staff are trained for their defect prevention roles.
2. The task kick-off meetings and causal analysis meetings are properly conducted.
3. The process for reviewing action proposals and implementing action items is followed.

Technology Change Management

a key process area for Level 5: Optimizing

The purpose of Technology Change Management is to identify new technologies (i.e., tools, methods, and processes) and track them into the organization in an orderly manner.

Technology Change Management involves identifying, selecting, and evaluating new technologies, and incorporating effective technologies into the organization. The objective is to improve software quality, increase productivity, and decrease the cycle time for product development.

The organization establishes a group (such as a software engineering process group or a technology support group) that works with the software projects to introduce and evaluate new technologies and manage changes to existing technologies. Particular emphasis is placed on technology changes that are likely to improve the capability of the organization's standard software process (as described in the Organization Process Definition key process area).

By maintaining an awareness of software-related technology innovations and systematically evaluating and experimenting with them, the organization selects appropriate technologies to improve the quality of its software and the productivity of its software activities. Pilot efforts are performed to assess new and unproven technologies before they are incorporated into normal practice. With appropriate sponsorship of the organization's management, the selected technologies are incorporated

into the organization's standard software process and current projects, as appropriate.

Changes to the organization's standard software process (as described in the Organization Process Definition key process area) and the projects' defined software processes (as described in the Integrated Software Management key process area) resulting from these technology changes are handled as described in the Process Change Management key process area.

Goals

- | | |
|---------------|---|
| Goal 1 | Incorporation of technology changes are planned. |
| Goal 2 | New technologies are evaluated to determine their effect on quality and productivity. |
| Goal 3 | Appropriate new technologies are transferred into normal practice across the organization. |

Commitment to perform

- | | |
|---------------------|---|
| Commitment 1 | The organization follows a written policy for improving its technology capability. |
|---------------------|---|

This policy typically specifies that:

1. Objectives for technology change management are established and documented.

- (Commitment 1) 2. A documented plan addresses the objectives for technology change management.

Commitment 2 Senior management sponsors the organization's activities for technology change management.

Senior management:

1. Helps to define a strategy that addresses the organization's goals for product quality, productivity, and cycle time for product development.
2. Helps to define a strategy that addresses the customer's and end users' needs and desires, as appropriate.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

3. Coordinates with the organization's managers in defining their goals and approaches for accomplishing the organization's strategy.
4. Makes a commitment to the effort for technology change management that is visible throughout the organization.
5. Establishes long-term plans and commitments for funding, staffing, and other resources.

Commitment 3 Senior management oversees the organization's technology change management activities.

Senior management:

1. Helps to establish policies for technology change management and reviews and approves these policies.
2. Allocates resources for technology change management activities.

- (Commitment 3) 3. Helps relate organizational strategies and objectives to strategies for technology change management.
4. Participates in establishing the plans for technology change management.
- ☐ Senior management coordinates requirements and issues for technology change management at all appropriate levels of the organization.
 - ☐ Senior management coordinates with the organization's managers to secure the managers' and staff's support and participation.

Ability to perform

Ability 1

A group responsible for the organization's technology change management activities exists.

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time. Considerations when implementing a group include assigned tasks or activities, the size of the project, the organizational structure, and the organizational culture. Some groups, such as the software quality assurance group, are focused on project activities, and others, such as the software engineering process group, are focused on organization-wide activities.

1. The group is either part of the group responsible for the organization's software process activities (e.g., software engineering process group) or its activities are closely coordinated with that group.
2. The group coordinates and helps to:
 - ☐ explore potential areas for applying new technology;

(Ability 1)

- ☐ select and plan for new technologies;
- ☐ acquire, install, and customize new technologies;
- ☐ communicate and coordinate with related research and development activities within the organization; and
- ☐ communicate with the technology suppliers on problems and enhancements.

Ability 2

Adequate resources and funding are provided to establish and staff a group responsible for the organization's technology change management activities.

1. Experienced staff members with expertise in specialized areas are available to this group to help in evaluating, planning, and supporting initiatives for technology change management.

Examples of specialized areas include:

- workstations,
- computer hardware,
- software reuse,
- computer-aided software engineering (CASE) technology,
- software measurement,
- formal methods, and
- programming languages.

2. Tools to support technology change management are made available.

Examples of support tools include:

- workstations,
- database programs, and
- subscriptions to on-line technology databases.

Ability 3

Support exists for collecting and analyzing data needed to evaluate technology changes.

(Ability 3)

This support includes the ability to:

1. Record selected process and product data automatically.
2. Support data analysis.
3. Display selected data.

The results of data analysis are presented in formats that appropriately convey the information content, e.g., graphical displays.

Ability 4

Appropriate data on the software processes and software work products are available to support analyses performed to evaluate and select technology changes.

Examples of process and product data include:

- resource expenditures and productivity by project, process stage, tools and methods used, program category, degree of program modification, etc.;
- schedule time by project, process stage of each project, program category, program size, degree of program modification, etc.;
- peer-review data, including defect data and review efficiencies;
- defect data showing stage introduced, stage removed, type, cause, severity, and time and effort to fix;
- change activity, including amount of code produced, amount of documentation produced, etc.;
- data on the activities to fix defects, including the identification of the defects, the product version where the defect fix was implemented, and identification of defects introduced in implementing each defect fix; and
- density of defects by project, product type, specific product, and specific subproduct (e.g., program modules).

Ability 5

Members of the group responsible for the organization's technology change management activities receive required training to perform these activities.

Examples of training include:

- the organization's standard software process,
- technology transfer and change management,
- software process improvement,
- tools and methods used by the organization,
- analytical and support facilities available to the organization, and
- principles of statistical quality control.

Refer to the Training Program key process area.

Activities performed

Activity 1

The organization develops and maintains a plan for technology change management.

This plan:

1. Covers the assigned responsibilities and resources required, including staff and tools.
2. Defines the long-term technical strategy for automating and improving the organization's standard software process and enhancing the organization's market position.
3. Identifies the procedures to be followed in performing the organization's technology change management activities.

(Activity 1)

4. Describes the approach for introducing new technologies to address specific needs of the organization and projects.
 - ☐ Process areas that are potential areas for technology changes are identified.
 - ☐ Approaches for identifying opportunities for technology changes are identified.
 - ☐ The specific planned or candidate technologies are identified.
 - ☐ Where appropriate, the life span for the planned technologies is estimated, from introduction to replacement.
 - ☐ The make/buy tradeoff studies are documented.
 - ☐ Approaches for assessing unproven candidate technologies are defined.
 - ☐ The acquisition and installation procedures are defined.
 - ☐ The initial training, continuing training, and consultation support are defined.
5. Undergoes peer review.

Refer to the Peer Reviews key process area.

6. Is reviewed by the affected managers.

Activity 2

The group responsible for the organization's technology change management activities works with the software projects in identifying areas of technology change.

This group:

1. Solicits suggestions for technology changes.
2. Identifies available new technologies that may be appropriate to the organization's and projects' needs.
 - ☐ A periodic search is made to identify commercially available technologies that meet identified and anticipated needs.

(Activity 2)

- ☐ Systematic efforts are made to maintain awareness of leading relevant technical work and trends of new technologies.
 - ☐ Systematic efforts are made to review the technologies used externally and to compare these technologies to those used within the organization.
 - ☐ Areas where new technologies have been used successfully are identified, and data and documentation of experience with using them are collected and reviewed.
3. Evaluates new technologies to determine their applicability to the organization's and projects' current and future needs.

Activity 3

Software managers and technical staff are kept informed of new technologies.

1. Information on new technologies is disseminated as appropriate.
2. Information on advanced technologies already in use in parts of the organization is disseminated as appropriate.
3. Information on the status of technologies being transferred into the organization is disseminated as appropriate.

Activity 4

The group responsible for the organization's technology change management systematically analyzes the organization's standard software process to identify areas that need or could benefit from new technology.

This group:

1. Analyzes the organization's standard software process to determine areas where new technologies would be most helpful.
2. Identifies helpful technology changes and determines the economics of those changes.
3. Defines the relationship of the identified technology to the organization's standard software process.

- (Activity 4)
4. Defines the expected outcomes of the technology change qualitatively and quantitatively, as appropriate.
 5. Determines the need for piloting each potential technology change.
 6. Determines the priority of the candidate new technologies.
 7. Documents results of the analysis activities.

Activity 5 Technologies are selected and acquired for the organization and software projects according to a documented procedure.

This procedure typically specifies that:

1. Requests for the acquisition of new technologies are documented.
 - ☐ Management approval is required for technologies with projected expenses above a predefined level.
2. Preliminary cost/benefit analyses are performed for the potential technology changes.
3. Predefined and approved selection criteria are used to identify the highest potential benefits.
4. Requirements and plans for the selected technology changes are defined and documented.
 - ☐ Where practical, the expected life span and plans for replacement/upgrade are estimated.
 - ☐ Where appropriate, tradeoff studies are performed, reviewed, and documented to determine whether the technology should be developed internally or procured externally.
 - ☐ Where appropriate, the plan provides for installing the new technology on a pilot basis to determine its effectiveness and economic benefits.
 - ☐ The requirements and plans are reviewed by the managers of the affected groups and the group responsible for technology change management activities.

Activity 6

Pilot efforts for improving technology are conducted, where appropriate, before a new technology is introduced into normal practice.

1. These pilot efforts are conducted to determine the feasibility and economics of untried or advanced technologies.
2. The plans for the pilot effort are documented.
 - ☐ The plan covers the objectives, evaluation criteria, and activities for the pilot effort.
3. The plan for conducting the pilot effort is reviewed and approved by the managers of the affected groups.

Examples of affected groups include:

- software engineering (including all subgroups),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

4. The group responsible for technology change management activities provides consultation and assistance to the project implementing the pilot effort.
5. The pilot effort is performed in an environment that is relevant to the development or maintenance environment.
6. The results of the pilot effort are collected, analyzed, and documented.
 - ☐ Lessons learned and problems encountered during the effort are documented.

(Activity 6)

- ☐ The benefits and impacts of broader use in the organization are estimated. The uncertainty in these estimates is assessed.
- ☐ A decision is made whether to terminate the effort, proceed with broad-scale implementation of the technology, or replan and continue the pilot effort.

Activity 7

Appropriate new technologies are incorporated into the organization's standard software process according to a documented procedure.

Refer to Activity 1 of the Organization Process Definition key process area and Activity 5 of the Process Change Management key process area for practices covering changes to the organization's standard software process.

Activity 8

Appropriate new technologies are incorporated into the projects' defined software processes according to a documented procedure.

Refer to Activity 2 of the Integrated Software Management key process area for practices covering revision of the project's defined software process.

Measurement and analysis

Measurement 1

Measurements are made and used to determine the status of the organization's activities for technology change management.

(Measurement 1)

Examples of measurements include:

- the overall technology change activity, including number, type, and size of changes; and
- the effect of implementing the technology change, compared to the goals.

Verifying implementation

Verification 1

The organization's activities for technology change management are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

These reviews:

1. Summarize the activities for technology change management.
2. Identify needed strategy changes.
3. Result in the resolution of issues.
4. Result in the approval of revisions to the plans for technology change management, as appropriate.

Verification 2

The software quality assurance group reviews and/or audits the activities and work products for technology change management and reports the results.

(Verification 2)

Refer to the Software Quality Assurance key process area.

At a minimum, the reviews and/or audits verify:

1. The plans for technology change management.
2. The process for selecting, procuring, and installing new technologies.

Process Change Management

a key process area for Level 5: Optimizing

The purpose of Process Change Management is to continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

Process Change Management involves defining process improvement goals and, with senior management sponsorship, proactively and systematically identifying, evaluating, and implementing improvements to the organization's standard software process and the projects' defined software processes on a continuous basis.

Training and incentive programs are established to enable and encourage everyone in the organization to participate in process improvement activities. Improvement opportunities are identified and evaluated for potential payback to the organization. Pilot efforts are performed to assess process changes before they are incorporated into normal practice.

When software process improvements are approved for normal practice, the organization's standard software process and the projects' defined software processes are revised as appropriate. The practices for revising the organization's standard software process are found in the Organization Process Definition key process area, and the practices for revising the projects' defined software processes are found in the Integrated Software Management key process area.

Goals

- Goal 1** **Continuous process improvement is planned.**
- Goal 2** **Participation in the organization's software process improvement activities is organization wide.**
- Goal 3** **The organization's standard software process and the projects' defined software processes are improved continuously.**

Commitment to perform

- Commitment 1** **The organization follows a written policy for implementing software process improvements.**

This policy typically specifies that:

1. The organization has quantitative, measurable goals for software process improvement and tracks performance against these goals.
2. The organization's process improvements are directed toward improving product quality, increasing productivity, and decreasing the cycle time for product development.
3. All of the organization's staff and managers are expected to participate in improving the software processes.

Skilled and motivated people are recognized as the principal process improvement resource.

- Commitment 2** **Senior management sponsors the organization's activities for software process improvement.**

(Commitment 2) Senior management:

1. Establishes the organization's long-term goals and plans for process improvement.
2. Allocates resources for process improvement activities.
3. Coordinates with the software managers to ensure they have reasonable, yet aggressive, process improvement goals and effective process improvement plans to meet these goals.
4. Monitors process improvement performance against goals.
5. Maintains a consistent priority focus on process improvement in the face of product crises.
6. Ensures that process improvement issues are promptly resolved.
7. Rewards employee participation in the process improvement activities.

Ability to perform

Ability 1

Adequate resources and funding are provided for software process improvement activities.

1. Resources are allocated to:
 - ☐ lead, guide, and support the process improvement activities;
 - ☐ maintain the process improvement records;
 - ☐ develop, control, and disseminate process changes; and
 - ☐ establish and operate the administrative and human resources functions to conduct the communications, motivation, and recognition activities needed to maintain a high level of employee participation.

(Ability 1)

2. Experienced individuals who have expertise in defining and analyzing software processes are available to help the organization in its process improvement activities.
3. Tools to support process improvement are made available.

Examples of support tools include:

- statistical analysis tools,
- database systems,
- process automation tools, and
- process modeling tools.

Ability 2

Software managers receive required training in software process improvement.

Examples of training include:

- managing technological and organizational change,
- team building, and
- teamwork skills as applied to continuous process improvement.

Refer to the Training Program key process area.

Ability 3

The managers and technical staff of the software engineering group and other software-related groups receive required training in software process improvement.

(Ability 3)

Examples of software-related groups include:

- software quality assurance,
- software configuration management, and
- documentation support.

Examples of training include:

- the principles of quality and process improvement, and
- the procedures for proposing process improvements.

Refer to the Training Program key process area.

Ability 4

Senior management receives required training in software process improvement.

Examples of training include:

- benchmarking and comparative evaluation,
- principles of process improvement,
- setting and tracking goals for process improvement, and
- motivation and team building in an environment of continuous process improvement.

Activities performed

Activity 1

A software process improvement program is established which empowers the members of the organization to improve the processes of the organization.

Activity 2

The group responsible for the organization's software process activities (e.g., software engineering process group) coordinates the software process improvement activities.

Refer to the Organization Process Focus key process area for practices covering the group responsible for the organization's software process improvement activities.

This group:

1. Defines organizational goals and measurement plans for software process performance.
2. Reviews the organizational goals for process performance with senior management for their endorsement.
3. Participates in the effort to define the organization's training needs for process improvement and supports the development and presentation of training course materials.

Refer to the Training Program key process area.

4. Defines and maintains the procedures for handling process improvement proposals.
5. Reviews software process improvement proposals and coordinates the actions for these proposals.
6. Tracks status, accomplishments, and participation in the process improvement activities and periodically reports the results to senior management.
7. Coordinates and tracks changes to the organization's standard software process.
8. Defines, establishes, and maintains the process improvement records.

Activity 3

The organization develops and maintains a plan for software process improvement according to a documented procedure.

Refer to Activity 2 of the Organization Process Focus key process area for other practices covering the organization's software process improvement plan.

This procedure typically specifies that:

1. The software process improvement plan is based on:
 - ☐ the organization's business and strategic operating plans, and
 - ☐ customer satisfaction indicators.
2. The software process improvement plan undergoes peer review.

Refer to the Peer Reviews key process area.

3. The software process improvement plan is reviewed by the affected managers.
4. The software process improvement plan is managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

If a greater degree of control than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Activity 4

The software process improvement activities are performed in accordance with the software process improvement plan.

The plan covers:

1. The resources required, including staff and tools.
2. The highest priority process areas for improvement.
3. Measurable short-term and long-term goals for software process performance and improvement.
4. Teams and their assignments for addressing improvements for specific process areas.

Examples of teams include:

- working groups,
- process action teams, and
- technical committees.

5. The procedures for:
 - ☐ the senior managers overseeing the software process improvement activities;
 - ☐ the software managers planning and coordinating the software process improvement activities;
 - ☐ individuals and teams identifying, evaluating, and introducing appropriate software process improvements; and
 - ☐ the teams developing software process improvements for assigned process areas.
6. The administrative and support plans required to maintain continuous process improvement.
 - ☐ Appropriate administrative procedures are included to encourage participation in and facilitate the software process improvement activities.
 - ☐ Administrative personnel are included in oversight and review of the software process improvement activities.

(Activity 4)

- ☐ The roles and contributions of employees to continuous process improvement are recognized.

Activity 5

Software process improvement proposals are handled according to a documented procedure.

This procedure typically specifies that:

1. Software process improvement proposals are submitted.

The software process improvement proposals can be submitted at any time and can address any area of the software processes.

Examples of sources for software process improvement proposals include:

- the findings and recommendations of software process assessments,
- the organization's software process improvement goals,
- analysis of data on customer problems and customer satisfaction,
- analysis of data on project performance compared to software quality and productivity goals,
- the results of process benchmarks,
- the potential for process/task automation,
- analysis of data on defect causes,
- the measured effectiveness of the software process activities,
- examples of software process improvement proposals that were successfully adopted, and
- feedback on previously submitted software process improvement proposals, as appropriate.

2. Each software process improvement proposal is evaluated; a decision is made whether to implement the proposal, and the decision rationale is documented.

(Activity 5)

3. The expected benefits of each software process improvement proposal are determined.

Examples of expected benefit areas include:

- productivity,
- quality,
- cycle time,
- other indicators of customer or end user satisfaction, and
- any other internal factors.

4. The priority of software process improvement proposals selected for implementation is determined.

- ☐ Focus on high-priority software process improvement proposals is maintained.

5. Implementation of the software process improvement actions resulting from the proposals is assigned and planned.

6. Software process improvement actions that require a substantial effort are assigned to a team responsible for implementation.

Examples of substantial efforts include improvements requiring piloting of new technologies and other large changes.

Teams to focus on specific software process areas are established.
Actions that are appropriate for piloting are coordinated.

Examples of teams include:

- working groups,
- process action teams, and
- technical committees.

(Activity 5)

7. The status of each software process improvement proposal is tracked.
8. Software process improvement proposals for which the response has been unusually long are identified and acted upon.
9. Software process changes that are judged to have a major impact on product quality or productivity or that will significantly alter satisfaction of the customer and end users are reviewed and approved by appropriate management before they are implemented.
10. Completed software process improvement actions are reviewed, verified, and approved before they are closed.
11. Submitters of the software process improvement proposals receive:
 - ☐ prompt acknowledgment of their proposals, and
 - ☐ notification of the disposition of their proposals.

Activity 6

Members of the organization actively participate in teams to develop software process improvements for assigned process areas.

1. Each of these process improvement teams is funded and the activities are planned and scheduled.
2. Goals are established for each process improvement effort; where possible, these goals are defined quantitatively.
3. The plans are approved by the managers of the affected groups and the group that defines and maintains the affected process descriptions.

(Activity 6)

Examples of affected groups include:

- software engineering (including all subgroups, such as software design, as well as the software task leaders),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- contract management, and
- documentation support.

Activity 7

Where appropriate, the software process improvements are installed on a pilot basis to determine their benefits and effectiveness before they are introduced into normal practice.

1. Adjustments to the proposed process improvement are made and documented during the pilot effort to optimize its implementation.
2. Lessons learned and problems encountered are documented.
3. The benefits, risks, and impacts of the process improvement's broader use in the organization are estimated, and the uncertainty in these estimates is assessed.
4. A decision is made whether to terminate the effort, proceed with broad-scale implementation of the improvement, or replan and continue the pilot effort.

Activity 8

When the decision is made to transfer a software process improvement into normal practice, the improvement is implemented according to a documented procedure.

This procedure typically specifies that:

1. The resources needed to support major changes to the software process are established and funded.

(Activity 8)

2. The strategy for collecting data to measure and track the change in software process performance is documented, reviewed, and agreed to.
 - ☐ This strategy is agreed to by the individuals responsible for implementing the software processes affected by the change.
 - ☐ The support tools are instrumented, as appropriate, to record the desired data automatically.
3. Training courses are updated to reflect the current software process, and training is provided before installing the process change for general use.

Refer to the Training Program key process area.

4. Consultation support, appropriate to the expected needs, is established before installing the process change for broad-scale use and is continued as needed.
5. Appropriate process changes are incorporated into the organization's standard software process.

Refer to Activity 1 of the Organization Process Definition key process area for practices covering the organization's standard software process.

6. Appropriate process changes are incorporated into the projects' defined software processes.

Refer to Activity 2 of the Integrated Software Management key process area for practices covering the project's defined software process.

Activity 9

Records of software process improvement activities are maintained.

(Activity 9)

1. Information about the initiation, status, and implementation of software process improvement proposals is maintained.
2. Ready access is provided to the software process improvement records.
3. Historical data is maintained and reports are produced on software process improvements.

Examples of records and reports include:

- the project's productivity, quality, and schedule performance;
- the program's defect history;
- the organizational software quality and productivity trends; and
- the cost, schedule, and productivity of software process development and improvement.

Refer to Activity 5 of the Organization Process Definition key process area for practices covering the organization's software process database, which is one of the possible mechanisms for maintaining process improvement records.

Activity 10

Software managers and technical staff receive feedback on the status and results of the software process improvement activities on an event-driven basis.

The feedback provides:

1. A summary of the major software process improvement activities.
2. Significant innovations and actions taken to address software process improvement.
3. A summary status of the software process improvement proposals that are submitted, open, and completed.

(Activity 10)

Examples of means to provide this feedback include:

- electronic bulletin boards,
- newsletters, and
- information flow meetings.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the software process improvement activities.**

Examples of measurements include:

- the number of software process improvement proposals submitted and implemented for each process area;
- the number of software process improvement proposals submitted by each of the projects, groups, and departments;
- the number and types of awards and recognitions received by each of the projects, groups, and departments;
- the response time for handling software process improvement proposals;
- the percentage of software process improvement proposals accepted per reporting period;
- the overall change activity, including number, type, and size of changes;
- the effect of implementing each process improvement compared to its defined goals;
- overall performance of the organization's and project's processes, including effectiveness, quality, and productivity compared to their defined goals;
- overall productivity and software quality trends for each project; and
- process measurements that relate to the indicators of the customer's satisfaction.

Verifying implementation

Verification 1

The activities for software process improvement are reviewed with senior management on a periodic basis.

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

These reviews are held to:

1. Summarize participation in the process improvement activities.
2. Assess process performance.
3. Identify needed goal changes.
4. Resolve issues.
5. Approve revisions to the software process improvement plan as appropriate.

Verification 2

The software quality assurance group reviews and/or audits the activities and work products for software process improvement and reports the results.

Refer to the Software Quality Assurance key process area.

(Verification 2) At a minimum, the reviews and/or audits verify:

1. The preparation of the organization's software process improvement plan.
2. The process of initiating, submitting, reviewing, approving, and planning implementation of software process improvement proposals.
3. The degree to which the process measurements conform to the software process descriptions and reflect actual performance.
4. The process for documenting, reviewing, approving, controlling, and disseminating changes to the organization's standard software process and projects' defined software processes.
5. The degree to which software process improvement activities are consistently measured and tracked.
6. The degree to which actual software process improvement performance achieves the plans and goals.

Capability Maturity Model

Appendices

	<i>Appendices</i>
Appendix A: References	A-1
Appendix B: Glossary of Terms	A-3
Appendix C: Abridged Version of the CMM Practices	A-27
Appendix D: Change History	A-67
Appendix E: Index	A-73

Appendix A: References

- Fagan86 M.E. Fagan, "Advances in Software Inspections," *IEEE Transactions on Software Engineering*, Vol. 12, No. 7, July, 1986, pp. 744-751.
- Feiler92 P.H. Feiler and W.S. Humphrey, *Software Process Development and Enactment: Concepts and Definitions*, CMU/SEI-92-TR-4, ADA258465, March 1992.
- Fowler90 P. Fowler and S. Rifkin, *Software Engineering Process Group Guide*, Software Engineering Institute, CMU/SEI-90-TR-24, ADA235784, September, 1990.
- Freedman90 D.P. Freedman and G.M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews, Third Edition*, Dorset House, New York, NY, 1990.
- Humphrey88 W.S. Humphrey, "Characterizing the Software Process," *IEEE Software*, Vol. 5, No. 2, March, 1988, pp. 73-79.
- Humphrey89 W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.
- IEEE-STD-610 ANSI/IEEE Std 610.12-1990, "IEEE Standard Glossary of Software Engineering Terminology," February 1991.
- Kitson92 D.H. Kitson and S. Masters, *An Analysis of SEI Software Process Assessment Results: 1987-1991*, Software Engineering Institute, CMU/SEI-92-TR-24, July 1992.

References

- Paulk91 M.C. Paulk, B. Curtis, M.B. Chrissis, et al, *Capability Maturity Model for Software*, Software Engineering Institute, CMU/SEI-91-TR-24, ADA240603, August 1991.
- Paulk93a M.C. Paulk, B. Curtis, M.B. Chrissis, and Charles V. Weber, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-24, February 1993.
- Paulk93b M.C. Paulk, C.V. Weber, S. Garcia, M.B. Chrissis, and M. Bush, *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-25, February 1993.
- Weber91 C.V. Weber, M.C. Paulk, C.J. Wise, and J.V. Withey, *Key Practices of the Capability Maturity Model*, Software Engineering Institute, CMU/SEI-91-TR-25, ADA240604, August 1991.

Appendix B: Glossary of Terms

ability to perform - (See *common features*.)

acceptance criteria - The criteria that a system or component must satisfy in order to be accepted by a user, customer, or other authorized entity. [IEEE-STD-610]

acceptance testing - Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. [IEEE-STD-610]

activity - Any step taken or function performed, both mental and physical, toward achieving some objective. Activities include all the work the managers and technical staff do to perform the tasks of the project and organization. (See *task* for contrast.)

activities performed - (See *common features*.)

action item- (1) A unit in a list that has been assigned to an individual or group for disposition. (2) An action proposal that has been accepted.

action proposal- A documented suggestion for change to a process or process-related item that will prevent the future occurrence of defects identified as a result of defect prevention activities. (See also *software process improvement proposal*.)

allocated requirements - (See *system requirements allocated to software*.)

application domain - A bounded set of related systems (i.e., systems that address a particular type of problem). Development and maintenance in an application domain usually requires special skills and/or resources. Examples include payroll and personnel systems, command and control systems, compilers, and expert systems.

assessment - (See *software process assessment*.)

Glossary of Terms

audit - An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria. [IEEE-STD-610]

baseline - A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. [IEEE-STD-610]

baseline configuration management- The establishment of baselines that are formally reviewed and agreed on and serve as the basis for further development. Some software work products, e.g., the software design and the code, should have baselines established at predetermined points, and a rigorous change control process should be applied these items. These baselines provide control and stability when interacting with the customer. (See also *baseline management*.)

baseline management- In configuration management, the application of technical and administrative direction to designate the documents and changes to those documents that formally identify and establish baselines at specific times during the life cycle of a configuration item. [IEEE-STD-610]

benchmark - A standard against which measurements or comparisons can be made. [IEEE-STD-610]

bidder - An individual, partnership, corporation, or association that has submitted a proposal and is a candidate to be awarded a contract to design, develop, and/or manufacture one or more products.

capability maturity model - A description of the stages through which software organizations evolve as they define, implement, measure, control, and improve their software processes. This model provides a guide for selecting process improvement strategies by facilitating the determination of current process capabilities and the identification of the issues most critical to software quality and process improvement.

causal analysis - The analysis of defects to determine their underlying root cause.

causal analysis meeting - A meeting, conducted after completing a specific task, to analyze defects uncovered during the performance of that task.

CMM - Acronym for *capability maturity model*.

commitment - A pact that is freely assumed, visible, and expected to be kept by all parties.

commitment to perform - (See *common features*.)

common cause (of a defect) - A cause of a defect that is inherently part of a process or system. Common causes affect every outcome of the process and everyone working in the process. (See *special cause* for contrast.)

common features - The subdivision categories of the CMM key process areas. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting. The CMM common features are the following:

- ❑ *commitment to perform* - The actions the organization must take to ensure that the process is established and will endure. Commitment to Perform typically involves establishing organizational policies and senior management sponsorship.
- ❑ *ability to perform* - The preconditions that must exist in the project or organization to implement the software process competently. Ability to Perform typically involves resources, organizational structures, and training.
- ❑ *activities performed* - A description of the roles and procedures necessary to implement a key process area. Activities Performed typically involve establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.
- ❑ *measurement and analysis* - A description of the need to measure the process and analyze the measurements. Measurement and Analysis typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed.

Glossary of Terms

- *verifying implementation* - The steps to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

configuration - In configuration management, the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product. [IEEE-STD-610]

configuration control - An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. [IEEE-STD-610]

configuration identification - An element of configuration management, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation. [IEEE-STD-610]

configuration item - An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process. [IEEE-STD-610]

configuration management - A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. [IEEE-STD-610]

configuration management library system - The tools and procedures to access the contents of the software baseline library.

configuration unit - The lowest level entity of a configuration item or component that can be placed into, and retrieved from, a configuration management library system.

Glossary of Terms

consistency - The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of system or component.
[IEEE-STD-610]

contingency factor - An adjustment (increase) of a size, cost, or schedule plan to account for likely underestimates of these parameters due to incomplete specification, inexperience in estimating the application domain, etc.

contract terms and conditions - The stated legal, financial, and administrative aspects of a contract.

critical computer resource - The parameters of the computing resources deemed to be a source of risk to the project because the potential need for those resources may exceed the amount that is available. Examples include target computer memory and host computer disk space.

critical path - A series of dependent tasks for a project that must be completed as planned to keep the entire project on schedule.

customer - The individual or organization that is responsible for accepting the product and authorizing payment to the developing organization.

defect - A flaw in a system or system component that causes the system or component to fail to perform its required function. A defect, if encountered during execution, may cause a failure of the system.

defect density - The number of defects identified in a product divided by the size of the product component (expressed in standard measurement terms for that product).

defect prevention - The activities involved in identifying defects or potential defects and preventing them from being introduced into a product.

defect root cause - The underlying reason (e.g., process deficiency) that allowed a defect to be introduced.

Glossary of Terms

defined level - (See *maturity level*.)

defined software process - (See *project's defined software process*.)

dependency item - A product, action, piece of information, etc., that must be provided by one individual or group to a second individual or group so that the second individual or group can perform a planned task.

developmental configuration management - The application of technical and administrative direction to designate and control software and associated technical documentation that define the evolving configuration of a software work product during development. Developmental configuration management is under the direct control of the developer. Items under developmental configuration management are not baselines, although they may be baselined and placed under baseline configuration management at some point in their development.

deviation - A noticeable or marked departure from the appropriate norm, plan, standard, procedure, or variable being reviewed.

documented procedure - (See *procedure*.)

effective process - A process that can be characterized as practiced, documented, enforced, trained, measured, and able to improve. (See also *well-defined process*.)

end user - The individual or group who will use the system for its intended operational use when it is deployed in its environment.

end user representatives - A selected sample of end users who represent the total population of end users.

engineering group - A collection of individuals (both managers and technical staff) representing an engineering discipline. Examples of engineering disciplines include systems engineering, hardware engineering, system test, software engineering, software configuration management, and software quality assurance.

evaluation - (See *software capability evaluation*.)

event-driven review/activity - A review or activity that is performed based on the occurrence of an event within the project (e.g., a formal review or the completion of a life cycle stage). (See *periodic review/activity* for contrast.)

findings - The conclusions of an assessment, evaluation, audit, or review that identify the most important issues, problems, or opportunities within the area of investigation.

first-line software manager - A manager who has direct management responsibility (including providing technical direction and administering the personnel and salary functions) for the staffing and activities of a single organizational unit (e.g., a department or project team) of software engineers and other related staff.

formal review - A formal meeting at which a product is presented to the end user, customer, or other interested parties for comment and approval. It can also be a review of the management and technical activities and of the progress of the project.

function - A set of related actions, undertaken by individuals or tools that are specifically assigned or fitted for their roles, to accomplish a set purpose or end.

goals - A summary of the key practices of a key process area that can be used to determine whether an organization or project has effectively implemented the key process area. The goals signify the scope, boundaries, and intent of each key process area.

group - The collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time.

Glossary of Terms

host computer - A computer used to develop software. (See *target computer* for contrast.)

initial level - (See *maturity level*.)

institutionalization - The building of infrastructure and corporate culture that support methods, practices, and procedures so that they are the ongoing way of doing business, even after those who originally defined them are gone.

integrated software management - The unification and integration of the software engineering and management activities into a coherent defined software process based on the organization's standard software process and related process assets.

integration - (See *software integration*.)

key practices - The infrastructures and activities that contribute most to the effective implementation and institutionalization of a key process area.

key process area - A cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability. The key process areas have been defined to reside at a single maturity level. They are the areas identified by the SEI to be the principal building blocks to help determine the software process capability of an organization and understand the improvements needed to advance to higher maturity levels. The Level 2 key process areas in the CMM are Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Subcontract Management, Software Quality Assurance, and Software Configuration Management. The Level 3 key process areas in the CMM are Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Intergroup Coordination, and Peer Reviews. The Level 4 key process areas are Quantitative Process Management and Software Quality Management. The Level 5 key process areas are Defect Prevention, Technology Change Management, and Process Change Management.

life cycle - (See *software life cycle*.)

maintenance - The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment. [IEEE-STD-610]

managed and controlled - The process of identifying and defining software work products that are not part of a baseline and, therefore, are not placed under configuration management but that must be controlled for the project to proceed in a disciplined manner. "Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

managed level - (See *maturity level*.)

manager - A role that encompasses providing technical and administrative direction and control to individuals performing tasks or activities within the manager's area of responsibility. The traditional functions of a manager include planning, resourcing, organizing, directing, and controlling work within an area of responsibility.

maturity level - A well-defined evolutionary plateau toward achieving a mature software process. The five maturity levels in the SEI's Capability Maturity Model are:

- ☐ *initial* - The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
- ☐ *repeatable* - Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- ☐ *defined* - The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

Glossary of Terms

- ❑ *managed* - Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
- ❑ *optimizing* - Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

maturity questionnaire - A set of questions about the software process that sample the key practices in each key process area of the CMM. The maturity questionnaire is used as a springboard to appraise the capability of an organization or project to execute a software process reliably.

measure - A unit of measurement (such as source lines of code or document pages of design).

measurement - The dimension, capacity, quantity, or amount of something (e.g., 300 source lines of code or 7 document pages of design).

method - A reasonably complete set of rules and criteria that establish a precise and repeatable way of performing a task and arriving at a desired result.

methodology - A collection of methods, procedures, and standards that defines an integrated synthesis of engineering approaches to the development of a product.

milestone - A scheduled event for which some individual is accountable and that is used to measure progress.

nontechnical requirements - Agreements, conditions and/or contractual terms that affect and determine the management activities of a software project.

operational software - The software that is intended to be used and operated in a system when it is delivered to its customer and deployed in its intended environment.

optimizing level - (See *maturity level*.)

organization - A unit within a company or other entity (e.g., government agency or branch of service) within which many projects are managed as a whole. All projects within an organization share a common top-level manager and common policies.

organization's measurement program - The set of related elements for addressing an organization's measurement needs. It includes the definition of organization-wide measurements, methods and practices for collecting organizational measurement data, methods and practices for analyzing organizational measurement data, and measurement goals for the organization.

organization's software process assets - A collection of entities, maintained by an organization, for use by projects in developing, tailoring, maintaining, and implementing their software processes. These software process assets typically include:

- ☐ the organization's standard software process,
- ☐ descriptions of the software life cycles approved for use,
- ☐ the guidelines and criteria for tailoring the organization's standard software process,
- ☐ the organization's software process database, and
- ☐ a library of software process-related documentation.

Any entity that the organization considers useful in performing the activities of process definition and maintenance could be included as a process asset.

organization's software process database - A database established to collect and make available data on the software processes and resulting software work products, particularly as they relate to the organization's standard software process. The database contains or references both the actual measurement data and the related information needed to understand the measurement data and assess it for reasonableness and applicability. Examples of process and work product data include estimates of software size, effort, and cost; actual data on software size, effort, and cost; productivity data; peer review coverage and efficiency; and number and severity of defects found in the software code.

Glossary of Terms

organization's standard software process - The operational definition of the basic process that guides the establishment of a common software process across the software projects in an organization. It describes the fundamental software process elements that each software project is expected to incorporate into its defined software process. It also describes the relationships (e.g., ordering and interfaces) between these software process elements.

orientation - An overview or introduction to a topic for those overseeing or interfacing with the individuals responsible for performing in the topic area. (See *train* for contrast.)

Pareto analysis - The analysis of defects by ranking causes from most significant to least significant. Pareto analysis is based on the principle, named after the 19th-century economist Vilfredo Pareto, that most effects come from relatively few causes, i.e., 80% of the effects come from 20% of the possible causes.

peer review - A review of a software work product, following defined procedures, by peers of the producers of the product for the purpose of identifying defects and improvements.

peer review leader - An individual specifically trained and qualified to plan, organize, and lead a peer review.

periodic review/activity - A review or activity that occurs at specified regular time intervals. (See *event-driven review/activity* for contrast.)

policy - A guiding principle, typically established by senior management, which is adopted by an organization or project to influence and determine decisions.

prime contractor - An individual, partnership, corporation, or association that administers a subcontract to design, develop, and/or manufacture one or more products.

procedure - A written description of a course of action to be taken to perform a given task. [IEEE-STD-610]

process - A sequence of steps performed for a given purpose; for example, the software development process. [IEEE-STD-610]

process capability - The range of expected results that can be achieved by following a process. (See *process performance* for contrast.)

process capability baseline - A documented characterization of the range of expected results that would normally be achieved by following a specific process under typical circumstances. A process capability baseline is typically established at an organizational level. (See *process performance baseline* for contrast.)

process database - (See *organization's software process database*.)

process description - The operational definition of the major components of a process. Documentation that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a process. It may also include the procedures for determining whether these provisions have been satisfied. Process descriptions may be found at the task, project, or organizational level.

process development - The act of defining and describing a process. It may include planning, architecture, design, implementation, and validation.

process measurement - The set of definitions, methods, and activities used to take measurements of a process and its resulting products for the purpose of characterizing and understanding the process.

process performance - A measure of the actual results achieved by following a process. (See *process capability* for contrast.)

process performance baseline - A documented characterization of the actual results achieved by following a process, which is used as a benchmark for comparing actual process performance against expected process performance. A process performance baseline is typically established at the project level, although the initial process performance baseline will usually be derived from the process capability baseline. (See *process capability baseline* for contrast.)

Glossary of Terms

process tailoring - The activity of creating a process description by elaborating, adapting, and/or completing the details of process elements or other incomplete specifications of a process. Specific business needs for a project will usually be addressed during process tailoring.

product - (See *software product* and *software work product*.)

profile - A comparison, usually in graphical form, of plans or projections versus actuals, typically over time.

project - An undertaking requiring concerted effort, which is focused on developing and/or maintaining a specific product. The product may include hardware, software, and other components. Typically a project has its own funding, cost accounting, and delivery schedule.

project's defined software process - The operational definition of the software process used by a project. The project's defined software process is a well-characterized and understood software process, described in terms of software standards, procedures, tools, and methods. It is developed by tailoring the organization's standard software process to fit the specific characteristics of the project. (See also *organization's standard software process*, *effective process*, and *well-defined process*.)

project manager - The role with total business responsibility for an entire project; the individual who directs, controls, administers, and regulates a project building a software or hardware/software system. The project manager is the individual ultimately responsible to the customer.

project software manager - The role with total responsibility for all the software activities for a project. The project software manager is the individual the project manager deals with in terms of software commitments and who controls all the software resources for a project.

quality - (1) The degree to which a system, component, or process meets specified requirements. (2) The degree to which a system, component, or process meets customer or user needs or expectations. [IEEE-STD-610]

quality assurance - (See *software quality assurance*.)

quantitative control - Any quantitative or statistically-based technique appropriate to analyze a software process, identify special causes of variations in the performance of the software process, and bring the performance of the software process within well-defined limits.

repeatable level - (See *maturity level*.)

required training - Training designated by an organization to be required to perform a specific role.

risk - Possibility of suffering loss.

risk management - An approach to problem analysis which weighs risk in a situation by using risk probabilities to give a more accurate understanding of the risks involved. Risk management includes risk identification, analysis, prioritization, and control.

risk management plan - The collection of plans that describe the risk management activities to be performed on a project.

role - A unit of defined responsibilities that may be assumed by one or more individuals.

SCE - Acronym for *software capability evaluation*.

SCM - Acronym for *software configuration management*.

senior manager - A management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects.

software architecture - The organizational structure of the software or module. [IEEE-STD-610]

Glossary of Terms

software baseline audit - An examination of the structure, contents, and facilities of the software baseline library to verify that baselines conform to the documentation that describes the baselines.

software baseline library - The contents of a repository for storing configuration items and the associated records.

software build - An operational version of a software system or component that incorporates a specified subset of the capabilities the final software system or component will provide. [IEEE-STD-610]

software capability evaluation - An appraisal by a trained team of professionals to identify contractors who are qualified to perform the software work or to monitor the state of the software process used on an existing software effort.

software configuration control board - A group responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes.

software development plan - The collection of plans that describe the activities to be performed for the software project. It governs the management of the activities performed by the software engineering group for a software project. It is not limited to the scope of any particular planning standard, such as DOD-STD-2167A and IEEE-STD-1058, which may use similar terminology.

software engineering group - The collection of individuals (both managers and technical staff) who have responsibility for software development and maintenance activities (i.e., requirements analysis, design, code, and test) for a project. Groups performing software-related work, such as the software quality assurance group, the software configuration management group, and the software engineering process group, are not included in the software engineering group.

software engineering process group - A group of specialists who facilitate the definition, maintenance, and improvement of the software process used by

the organization. In the key practices, this group is generically referred to as "the group responsible for the organization's software process activities."

software engineering staff - The software technical people (e.g., analysts, programmers, and engineers), including software task leaders, who perform the software development and maintenance activities for the project, but who are not managers.

software integration - A process of putting together selected software components to provide the set or specified subset of the capabilities the final software system will provide.

software life cycle - The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase. [IEEE-STD-610]

software manager - Any manager, at a project or organizational level, who has direct responsibility for software development and/or maintenance.

software plans - The collection of plans, both formal and informal, used to express how software development and/or maintenance activities will be performed. Examples of plans that could be included: software development plan, software quality assurance plan, software configuration management plan, software test plan, risk management plan, and process improvement plan.

software process - A set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals).

software process assessment - An appraisal by a trained team of software professionals to determine the state of an organization's current software process, to determine the high-priority software process-related issues facing

Glossary of Terms

an organization and to obtain the organizational support for software process improvement.

software process assets - (See *organization's software process assets*.)

software process capability - (See *process capability*.)

software process description - The operational definition of a major software process component identified in the project's defined software process or the organization's standard software process. It documents, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a software process. (See also *process description*.)

software process element - A constituent element of a software process description. Each process element covers a well-defined, bounded, closely related set of tasks (e.g., software estimating element, software design element, coding element, and peer review element). The descriptions of the process elements may be templates to be filled in, fragments to be completed, abstractions to be refined, or complete descriptions to be modified or used unmodified.

software process improvement plan - A plan, derived from the recommendations of a software process assessment, that identifies the specific actions that will be taken to improve the software process and outlines the plans for implementing those actions. Sometimes referred to as an action plan.

software process improvement proposal - A documented suggestion for change to a process or process-related item that will improve software process capability and performance. (See also *action proposal*.)

software process maturity - The extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization.

software process performance - (See *process performance*.)

software process-related documentation - Example documents and document fragments, which are expected to be of use to future projects when they are tailoring the organization's standard software process. The examples may cover subjects such as a project's defined software process, standards, procedures, software development plans, measurement plans, and process training materials.

software product - The complete set, or any of the individual items of the set, of computer programs, procedures, and associated documentation and data designated for delivery to a customer or end user. [IEEE-STD-610] (See *software work product* for contrast.)

software project - An undertaking requiring concerted effort, which is focused on analyzing, specifying, designing, developing, testing, and/or maintaining the software components and associated documentation of a system. A software project may be part of a project building a hardware/software system.

software quality assurance - (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that a software work product conforms to established technical requirements. (2) A set of activities designed to evaluate the process by which software work products are developed and/or maintained.

software quality goal - Quantitative quality objectives defined for a software work product.

software quality management - The process of defining quality goals for a software product, establishing plans to achieve these goals, and monitoring and adjusting the software plans, software work products, activities, and quality goals to satisfy the needs and desires of the customer and end users.

software-related group - A collection of individuals (both managers and technical staff) representing a software engineering discipline that supports, but is not directly responsible for, software development and/or maintenance. Examples of software engineering disciplines include software quality assurance and software configuration management.

Glossary of Terms

software requirement - A condition or capability that must be met by software needed by a user to solve a problem or achieve an objective. [IEEE-STD-610]

software work product - Any artifact created as part of defining, maintaining, or using a software process, including process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to a customer or end user. (See *software product* for contrast.)

SPA - Acronym for *software process assessment*.

special cause (of a defect) - A cause of a defect that is specific to some transient circumstance and not an inherent part of a process. Special causes provide random variation (noise) in process performance. (See *common cause* for contrast.)

SQA - Acronym for *software quality assurance*.

staff - The individuals, including task leaders, who are responsible for accomplishing an assigned function, such as software development or software configuration management, but who are not managers.

stage - A partition of the software effort that is of a manageable size and that represents a meaningful and measurable set of related tasks which are performed by the project. A stage is usually considered a subdivision of a software life cycle and is often ended with a formal review prior to the onset of the following stage.

standard - Mandatory requirements employed and enforced to prescribe a disciplined uniform approach to software development.

standard software process - (See *organization's standard software process*.)

statement of work - A description of all the work required to complete a project, which is provided by the customer.

subcontract manager - A manager in the prime contractor's organization who has direct responsibility for administering and managing one or more subcontracts.

subcontractor - An individual, partnership, corporation, or association that contracts with an organization (i.e., the prime contractor) to design, develop, and/or manufacture one or more products.

system - A collection of components organized to accomplish a specific function or set of functions. [IEEE-STD-610]

system engineering group - The collection of individuals (both managers and technical staff) who have responsibility for specifying the system requirements; allocating the system requirements to the hardware, software, and other components; specifying the interfaces between the hardware, software, and other components; and monitoring the design and development of these components to ensure conformance with their specifications.

system requirement - A condition or capability that must be met or possessed by a system or system component to satisfy a condition or capability needed by a user to solve a problem. [IEEE-STD-610]

system requirements allocated to software - The subset of the system requirements that are to be implemented in the software components of the system. The allocated requirements are a primary input to the software development plan. Software requirements analysis elaborates and refines the allocated requirements and results in software requirements which are documented.

tailor - To modify a process, standard, or procedure to better match process or product requirements.

target computer - The computer on which delivered software is intended to operate. (See *host computer* for contrast.)

task - (1) A sequence of instructions treated as a basic unit of work. [IEEE-STD-610] (2) A well-defined unit of work in the software process that

Glossary of Terms

provides management with a visible checkpoint into the status of the project. Tasks have readiness criteria (preconditions) and completion criteria (postconditions). (See *activity* for contrast.)

task kick-off meeting - A meeting held at the beginning of a task of a project for the purpose of preparing the individuals involved to perform the activities of that task effectively.

task leader - The leader of a technical team for a specific task, who has technical responsibility and provides technical direction to the staff working on the task.

team - A collection of people, often drawn from diverse but related groups, assigned to perform a well-defined function for an organization or a project. Team members may be part-time participants of the team and have other primary responsibilities.

testability - (1) The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met. (2) The degree to which a requirement is stated in terms that permit establishment of test criteria and performance of tests to determine whether those criteria have been met. [IEEE-STD-610]

technical requirements - Those requirements that describe what the software must do and its operational constraints. Examples of technical requirements include functional, performance, interface, and quality requirements.

technology - The application of science and/or engineering in accomplishing some particular result.

traceability - The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another. [IEEE-STD-610]

train - To make proficient with specialized instruction and practice. (See also *orientation*.)

training group - The collection of individuals (both managers and staff) who are responsible for coordinating and arranging the training activities for an organization. This group typically prepares and conducts most of the training courses and coordinates use of other training vehicles.

training program - The set of related elements that focus on addressing an organization's training needs. It includes an organization's training plan, training materials, development of training, conduct of training, training facilities, evaluation of training, and maintenance of training records.

training waiver - A written approval exempting an individual from training that has been designated as required for a specific role. The exemption is granted because it has been objectively determined that the individual already possesses the needed skills to perform the role.

unit - (1) A separately testable element specified in the design of a computer software component. (2) A logically separable part of a computer program. (3) A software component that is not subdivided into other components. [IEEE-STD-610]

user- (See *end user*.)

validation- The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [IEEE-STD-610]

verification- The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610]

verifying implementation - (See *common features*.)

waiver - (See *training waiver*).

Glossary of Terms

well-defined process - A process that includes readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and completion criteria. (See also *effective process*.)

Appendix C: Abridged Version of the Key Practices

This appendix provides an abridged version of the key practices, which provides a high-level overview of the primary activities the SEI prescribes for each key process area. It can be used to get a "quick look" at each key process area. It does not, however, provide the specific activities for these key practices nor does it cover all the key practices. It is intended for informational purposes, not for determining compliance to the key practices or planning process improvements.

This abridgement contains a short description of the key process area, its goals, and the key practice statements from the Activities Performed common feature of the key process area. These items are extracted verbatim from the detailed key practice tables.

There are a number of other key practices specified under the other common features (i.e., Commitment to Perform, Ability to Perform, Measurement and Analysis, and Verifying Implementation) that are not contained in this appendix. These other key practices must be in place to ensure the key practices are implemented appropriately and effectively, are solidly established, will be maintained and not erode over time, and can be effectively applied to new work. To appropriately establish a key process area, the full set of key practices should be used.

Commitment to Perform typically involves establishing organizational policies and senior management sponsorship. Ability to Perform typically involves resources, organizational structures, and training. Measurement and Analysis typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed. Verifying Implementation typically encompasses reviews and audits by management and software quality assurance.

Level 2: Requirements Management

The purpose of Requirements Management is to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project.

Requirements Management involves establishing and maintaining an agreement with the customer on the requirements for the software project. This agreement is referred to as the "system requirements allocated to the software." The "customer" may be interpreted as the system engineering group, the marketing group, another internal organization, or an external customer. The agreement covers both the technical and nontechnical (e.g., delivery dates) requirements. The agreement forms the basis for estimating, planning, performing, and tracking the software project's activities throughout the software life cycle.

The allocation of the system requirements to software, hardware, and other system components (e.g., humans) may be performed by a group external to the software engineering group (e.g., the system engineering group), and the software engineering group may have no direct control of this allocation. Within the constraints of the project, the software engineering group takes appropriate steps to ensure that the system requirements allocated to software, which they are responsible for addressing, are documented and controlled.

To achieve this control, the software engineering group reviews the initial and revised system requirements allocated to software to resolve issues before they are incorporated into the software project. Whenever the system requirements allocated to software are changed, the affected software plans, work products, and activities are adjusted to remain consistent with the updated requirements.

The goals of Requirements Management are:

1. System requirements allocated to software are controlled to establish a baseline for software engineering and management use.
2. Software plans, products, and activities are kept consistent with the system requirements allocated to software.

The top-level activities performed for Requirements Management are:

1. The software engineering group reviews the allocated requirements before they are incorporated into the software project.
2. The software engineering group uses the allocated requirements as the basis for software plans, work products, and activities.
3. Changes to the allocated requirements are reviewed and incorporated into the software project.

Level 2: Software Project Planning

The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project.

Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work.

The software planning begins with a statement of the work to be performed and other constraints and goals that define and bound the software project (those established by the practices of the Requirements Management key process area). The software planning process includes steps to estimate the size of the software work products and the resources needed, produce a schedule, identify and assess software risks, and negotiate commitments. Iterating through these steps may be necessary to establish the plan for the software project (i.e., the software development plan).

This plan provides the basis for performing and managing the software project's activities and addresses the commitments to the software project's customer according to the resources, constraints, and capabilities of the software project.

The goals of Software Project Planning are:

1. Software estimates are documented for use in planning and tracking the software project.
2. Software project activities and commitments are planned and documented.
3. Affected groups and individuals agree to their commitments related to the software project.

The top-level activities performed for Software Project Planning are:

1. The software engineering group participates on the project proposal team.
2. Software project planning is initiated in the early stages of, and in parallel with, the overall project planning.
3. The software engineering group participates with other affected groups in the overall project planning throughout the project's life.
4. Software project commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.
5. A software life cycle with predefined stages of manageable size is identified or defined.
6. The project's software development plan is developed according to a documented procedure.
7. The plan for the software project is documented.
8. Software work products that are needed to establish and maintain control of the software project are identified.
9. Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.
10. Estimates for the software project's effort and costs are derived according to a documented procedure.
11. Estimates for the project's critical computer resources are derived according to a documented procedure.
12. The project's software schedule is derived according to a documented procedure.
13. The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented.
14. Plans for the project's software engineering facilities and support tools are prepared.
15. Software planning data are recorded.

Level 2: Software Project Tracking and Oversight

The purpose of Software Project Tracking and Oversight is to provide adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.

Software Project Tracking and Oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these plans based on the actual accomplishments and results.

A documented plan for the software project (i.e., the software development plan, as described in the Software Project Planning key process area) is used as the basis for tracking the software activities, communicating status, and revising plans. Software activities are monitored by the management. Progress is primarily determined by comparing the actual software size, effort, cost, and schedule to the plan when selected software work products are completed and at selected milestones. When it is determined that the software project's plans are not being met, corrective actions are taken. These actions may include revising the software development plan to reflect the actual accomplishments and replanning the remaining work or taking actions to improve the performance.

The goals of Software Project Tracking and Oversight are:

1. Actual results and performances are tracked against the software plans.
2. Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.
3. Changes to software commitments are agreed to by the affected groups and individuals.

The top-level activities performed for Software Project Tracking and Oversight are:

1. A documented software development plan is used for tracking the software activities and communicating status.
2. The project's software development plan is revised according to a documented procedure.
3. Software project commitments and changes to commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.
4. Approved changes to commitments that affect the software project are communicated to the members of the software engineering group and other software-related groups.
5. The size of the software work products (or size of the changes to the software work products) are tracked, and corrective actions are taken as necessary.
6. The project's software effort and costs are tracked, and corrective actions are taken as necessary.
7. The project's critical computer resources are tracked, and corrective actions are taken as necessary.
8. The project's software schedule is tracked, and corrective actions are taken as necessary.
9. Software engineering technical activities are tracked, and corrective actions are taken as necessary.
10. The software risks associated with cost, resource, schedule, and technical aspects of the project are tracked.
11. Actual measurement data and replanning data for the software project are recorded.
12. The software engineering group conducts periodic internal reviews to track technical progress, plans, performance, and issues against the software development plan.
13. Formal reviews to address the accomplishments and results of the software project are conducted at selected project milestones according to a documented procedure.

Level 2: Software Subcontract Management

The purpose of Software Subcontract Management is to select qualified software subcontractors and manage them effectively.

Software Subcontract Management involves selecting a software subcontractor, establishing commitments with the subcontractor, and tracking and reviewing the subcontractor's performance and results. These practices cover the management of a software (only) subcontract, as well as the management of the software component of a subcontract that includes software, hardware, and possibly other system components.

The subcontractor is selected based on its ability to perform the work. Many factors contribute to the decision to subcontract a portion of the prime contractor's work. Subcontractors may be selected based on strategic business alliances, as well as technical considerations. The practices of this key process area address the traditional acquisition process associated with subcontracting a defined portion of the work to another organization.

When subcontracting, a documented agreement covering the technical and nontechnical (e.g., delivery dates) requirements is established and is used as the basis for managing the subcontract. The work to be done by the subcontractor and the plans for the work are documented. The standards that are to be followed by the subcontractor are compatible with the prime contractor's standards.

The software planning, tracking, and oversight activities for the subcontracted work are performed by the subcontractor. The prime contractor ensures that these planning, tracking, and oversight activities are performed appropriately and that the software products delivered by the subcontractor satisfy their acceptance criteria. The prime contractor works with the subcontractor to manage their product and process interfaces.

The goals of Software Subcontract Management are:

1. The prime contractor selects qualified software subcontractors.
2. The prime contractor and the software subcontractor agree to their commitments to each other.
3. The prime contractor and the software subcontractor maintain ongoing communications.
4. The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

The top-level activities performed for Software Subcontract Management are:

1. The work to be subcontracted is defined and planned according to a documented procedure.
2. The software subcontractor is selected, based on an evaluation of the subcontract bidders' ability to perform the work, according to a documented procedure.
3. The contractual agreement between the prime contractor and the software subcontractor is used as the basis for managing the subcontract.
4. A documented subcontractor's software development plan is reviewed and approved by the prime contractor.
5. A documented and approved subcontractor's software development plan is used for tracking the software activities and communicating status.
6. Changes to the software subcontractor's statement of work, subcontract terms and conditions, and other commitments are resolved according to a documented procedure.
7. The prime contractor's management conducts periodic status/coordination reviews with the software subcontractor's management.

Abridged Practices

8. Periodic technical reviews and interchanges are held with the software subcontractor.
9. Formal reviews to address the subcontractor's software engineering accomplishments and results are conducted at selected milestones according to a documented procedure.
10. The prime contractor's software quality assurance group monitors the subcontractor's software quality assurance activities according to a documented procedure.
11. The prime contractor's software configuration management group monitors the subcontractor's activities for software configuration management according to a documented procedure.
12. The prime contractor conducts acceptance testing as part of the delivery of the subcontractor's software products according to a documented procedure.
13. The software subcontractor's performance is evaluated on a periodic basis, and the evaluation is reviewed with the subcontractor.

Level 2: Software Quality Assurance

The purpose of Software Quality Assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built.

Software Quality Assurance involves reviewing and auditing the software products and activities to verify that they comply with the applicable procedures and standards and providing the software project and other appropriate managers with the results of these reviews and audits.

The software quality assurance group works with the software project during its early stages to establish plans, standards, and procedures that will add value to the software project and satisfy the constraints of the project and the organization's policies. By participating in establishing the plans, standards, and procedures, the software quality assurance group helps ensure they fit the project's needs and verifies that they will be usable for performing reviews and audits throughout the software life cycle. The software quality assurance group reviews project activities and audits software work products throughout the life cycle and provides management with visibility as to whether the software project is adhering to its established plans, standards, and procedures.

Compliance issues are first addressed within the software project and resolved there if possible. For issues not resolvable within the software project, the software quality assurance group escalates the issue to an appropriate level of management for resolution.

This key process area covers the practices for the group performing the software quality assurance function. The practices identifying the specific activities and work products that the software quality assurance group reviews and/or audits are generally contained in the Verifying Implementation common feature of the other key process areas.

The goals of Software Quality Assurance are:

1. Software quality assurance activities are planned.
2. Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.
3. Affected groups and individuals are informed of software quality assurance activities and results.
4. Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

The top-level activities performed for Software Quality Assurance are:

1. A SQA plan is prepared for the software project according to a documented procedure.
2. The SQA group's activities are performed in accordance with the SQA plan.
3. The SQA group participates in the preparation and review of the project's software development plan, standards, and procedures.
4. The SQA group reviews the software engineering activities to verify compliance.
5. The SQA group audits designated software work products to verify compliance.
6. The SQA group periodically reports the results of its activities to the software engineering group.
7. Deviations identified in the software activities and software work products are documented and handled according to a documented procedure.
8. The SQA group conducts periodic reviews of its activities and findings with the customer's SQA personnel, as appropriate.

Level 2: Software Configuration Management

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

Software Configuration Management involves identifying the configuration of the software (i.e., selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products (e.g., the compiler).

A software baseline library is established containing the software baselines as they are developed. Changes to baselines and the release of software products built from the software baseline library are systematically controlled via the change control and configuration auditing functions of software configuration management.

This key process area covers the practices for performing the software configuration management function. The practices identifying specific configuration items/units are contained in the key process areas that describe the development and maintenance of each configuration item/unit.

The goals of Software Configuration Management are:

1. Software configuration management activities are planned.
2. Selected software work products are identified, controlled, and available.
3. Changes to identified software work products are controlled.
4. Affected groups and individuals are informed of the status and content of software baselines.

The top-level activities performed for Software Configuration Management are:

1. A SCM plan is prepared for each software project according to a documented procedure.
2. A documented and approved SCM plan is used as the basis for performing the SCM activities.
3. A configuration management library system is established as a repository for the software baselines.
4. The software work products to be placed under configuration management are identified.
5. Change requests and problem reports for all configuration items/units are initiated, recorded, reviewed, approved, and tracked according to a documented procedure.
6. Changes to baselines are controlled according to a documented procedure.
7. Products from the software baseline library are created and their release is controlled according to a documented procedure.
8. The status of configuration items/units is recorded according to a documented procedure.
9. Standard reports documenting the SCM activities and the contents of the software baseline are developed and made available to affected groups and individuals.
10. Software baseline audits are conducted according to a documented procedure.

Level 3: Organization Process Focus

The purpose of Organization Process Focus is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability.

Organization Process Focus involves developing and maintaining an understanding of the organization's and projects' software processes and coordinating the activities to assess, develop, maintain, and improve these processes.

The organization provides the long-term commitments and resources to coordinate the development and maintenance of the software processes across current and future software projects via a group such as a software engineering process group. This group is responsible for the organization's software process activities. It is specifically responsible for the development and maintenance of the organization's standard software process and related process assets (as described in the Organization Process Definition key process area), and it coordinates the process activities with the software projects.

The goals of Organization Process Focus are:

1. Software process development and improvement activities are coordinated across the organization.
2. The strengths and weaknesses of the software processes used are identified relative to a process standard.
3. Organization-level process development and improvement activities are planned.

The top-level activities performed for Organization Process Focus are:

1. The software process is assessed periodically, and action plans are developed to address the assessment findings.
2. The organization develops and maintains a plan for its software process development and improvement activities.
3. The organization's and projects' activities for developing and improving their software processes are coordinated at the organization level.
4. The use of the organization's software process database is coordinated at the organizational level.
5. New processes, methods, and tools in limited use in the organization are monitored, evaluated, and, where appropriate, transferred to other parts of the organization.
6. Training for the organization's and projects' software processes is coordinated across the organization.
7. The groups involved in implementing the software processes are informed of the organization's and projects' activities for software process development and improvement.

Level 3: Organization Process Definition

The purpose of Organization Process Definition is to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization.

Organization Process Definition involves developing and maintaining the organization's standard software process, along with related process assets, such as descriptions of software life cycles, process tailoring guidelines and criteria, the organization's software process database, and a library of software process-related documentation.

These assets may be collected in many ways, depending on the organization's implementation of Organization Process Definition. For example, the descriptions of the software life cycles may be an integral part of the organization's standard software process or parts of the library of software process-related documentation may be stored in the organization's software process database.

The organization's software process assets are available for use in developing, implementing, and maintaining the projects' defined software processes. (The practices related to the development and maintenance of the project's defined software process are described in the Integrated Software Management key process area.)

The goals of Organization Process Definition are:

1. A standard software process for the organization is developed and maintained.
2. Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

The top-level activities performed for Organization Process Definition are:

1. The organization's standard software process is developed and maintained according to a documented procedure.
2. The organization's standard software process is documented according to established organization standards.
3. Descriptions of software life cycles that are approved for use by the projects are documented and maintained.
4. Guidelines and criteria for the projects' tailoring of the organization's standard software process are developed and maintained.
5. The organization's software process database is established and maintained.
6. A library of software process-related documentation is established and maintained.

Level 3: Training Program

The purpose of the Training Program key process area is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently.

Training Program involves first identifying the training needed by the organization, projects, and individuals, then developing or procuring training to address the identified needs.

Each software project evaluates its current and future skill needs and determines how these skills will be obtained. Some skills are effectively and efficiently imparted through informal vehicles (e.g., on-the-job training and informal mentoring), whereas other skills need more formal training vehicles (e.g., classroom training and guided self-study) to be effectively and efficiently imparted. The appropriate vehicles are selected and used.

This key process area covers the practices for the group performing the training function. The practices identifying the specific training topics (i.e., knowledge or skill needed) are contained in the Ability to Perform common feature of the individual key process areas.

The goals of Training Program are:

1. Training activities are planned.
2. Training for developing the skills and knowledge needed to perform software management and technical roles is provided.
3. Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.

The top-level activities performed for Training Program are:

1. Each software project develops and maintains a training plan that specifies its training needs.
2. The organization's training plan is developed and revised according to a documented procedure.
3. The training for the organization is performed in accordance with the organization's training plan.
4. Training courses prepared at the organization level are developed and maintained according to organization standards.
5. A waiver procedure for required training is established and used to determine whether individuals already possess the knowledge and skills required to perform in their designated roles.
6. Records of training are maintained.

Level 3: Integrated Software Management

The purpose of Integrated Software Management is to integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets, which are described in Organization Process Definition.

Integrated Software Management involves developing the project's defined software process and managing the software project using this defined software process. The project's defined software process is tailored from the organization's standard software process to address the specific characteristics of the project.

The software development plan is based on the project's defined software process and describes how the activities of the project's defined software process will be implemented and managed. The management of the software project's size, effort, cost, schedule, staffing, and other resources is tied to the tasks of the project's defined software process.

Since the projects' defined software processes are all tailored from the organization's standard software process, the software projects can share process data and lessons learned.

The basic practices for estimating, planning, and tracking a software project are described in the Software Project Planning and Software Project Tracking and Oversight key process areas. They focus on recognizing problems when they occur and adjusting the plans and/or performance to address the problems. The practices of this key process area build on, and are in addition to, the practices of those two key process areas. The emphasis of Integrated Software Management shifts to anticipating problems and acting to prevent or minimize the effects of these problems.

The goals of Integrated Software Management are:

1. The project's defined software process is a tailored version of the organization's standard software process.
2. The project is planned and managed according to the project's defined software process.

The top-level activities performed for Integrated Software Management are:

1. The project's defined software process is developed by tailoring the organization's standard software process according to a documented procedure.
2. Each project's defined software process is revised according to a documented procedure.
3. The project's software development plan, which describes the use of the project's defined software process, is developed and revised according to a documented procedure.
4. The software project is managed in accordance with the project's defined software process.
5. The organization's software process database is used for software planning and estimating.
6. The size of the software work products (or size of changes to the software work products) is managed according to a documented procedure.
7. The project's software effort and costs are managed according to a documented procedure.
8. The project's critical computer resources are managed according to a documented procedure.
9. The critical dependencies and critical paths of the project's software schedule are managed according to a documented procedure.
10. The project's software risks are identified, assessed, documented, and managed according to a documented procedure.
11. Reviews of the software project are periodically performed to determine the actions needed to bring the software project's performance and results in line with the current and projected needs of the business, customer, and end users, as appropriate.

Level 3: Software Product Engineering

The purpose of Software Product Engineering is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.

Software Product Engineering involves performing the engineering tasks to build and maintain the software using the project's defined software process (which is described in the Integrated Software Management key process area) and appropriate method and tools.

The software engineering tasks include analyzing the system requirements allocated to software (these system requirements are described in the Requirements Management key process area), developing the software requirements, developing the software architecture, designing the software, implementing the software in the code, integrating the software components, and testing the software to verify that it satisfies the specified requirements (i.e., the system requirements allocated to software and the software requirements).

Documentation needed to perform the software engineering tasks (e.g., software requirements document, software design document, test plan, and test procedures) is developed and reviewed to ensure that each task addresses the results of predecessor tasks and the results produced are appropriate for the subsequent tasks (including the tasks of operating and maintaining the software). When changes are approved, affected software work products, plans, commitments, processes, and activities are revised to reflect the approved changes.

The goals of Software Product Engineering are:

1. The software engineering tasks are defined, integrated, and consistently performed to produce the software.
2. Software work products are kept consistent with each other.

The top-level activities performed for Software Product Engineering are:

1. Appropriate software engineering methods and tools are integrated into the project's defined software process.
2. The software requirements are developed, maintained, documented, and verified by systematically analyzing the allocated requirements according to the project's defined software process.
3. The software design is developed, maintained, documented, and verified, according to the project's defined software process, to accommodate the software requirements and to form the framework for coding.
4. The software code is developed, maintained, documented, and verified, according to the project's defined software process, to implement the software requirements and software design.
5. Software testing is performed according to the project's defined software process.
7. System and acceptance testing of the software are planned and performed to demonstrate that the software satisfies its requirements.
8. The documentation that will be used to operate and maintain the software is developed and maintained according to the project's defined software process.
9. Data on defects identified in peer reviews and testing are collected and analyzed according to the project's defined software process.
10. Consistency is maintained across software work products, including the software plans, process descriptions, allocated requirements, software requirements, software design, code, test plans, and test procedures.

Level 3: Intergroup Coordination

The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

Intergroup Coordination involves the software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues. Representatives of the project's engineering groups participate in establishing the system-level requirements, objectives, and plans by working with the customer and end users, as appropriate. These requirements, objectives, and plans become the basis for all engineering activities.

The technical working interfaces and interactions between groups are planned and managed to ensure the quality and integrity of the entire system. Technical reviews and interchanges are regularly conducted with representatives of the project's engineering groups to ensure that all engineering groups are aware of the status and plans of all the groups, and that system and intergroup issues receive appropriate attention.

The software-specific practices related to these engineering tasks are described in the Requirements Management and Software Product Engineering key process areas.

The goals of Intergroup Coordination are:

1. The customer's requirements are agreed to by all affected groups.
2. The commitments between the engineering groups are agreed to by the affected groups.
3. The engineering groups identify, track, and resolve intergroup issues.

The top-level activities performed for Intergroup Coordination are:

1. The software engineering group and the other engineering groups participate with the customer and end users, as appropriate, to establish the system requirements.
2. Representatives of the project's software engineering group work with representatives of the other engineering groups to monitor and coordinate technical activities and resolve technical issues.
3. A documented plan is used to communicate intergroup commitments and to coordinate and track the work performed.
4. Critical dependencies between engineering groups are identified, negotiated, and tracked according to a documented procedure.
5. Work products produced as input to other engineering groups are reviewed by representatives of the receiving groups to ensure that the work products meet their needs.
6. Intergroup issues not resolvable by the individual representatives of the project engineering groups are handled according to a documented procedure.
7. Representatives of the project engineering groups conduct periodic technical reviews and interchanges.

Level 3: Peer Reviews

The purpose of Peer Reviews is to remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of defects that might be prevented.

Peer Reviews involve a methodical examination of software work products by the producers' peers to identify defects and areas where changes are needed. The specific products that will undergo a peer review are identified in the project's defined software process and scheduled as part of the software project planning activities, as described in Integrated Software Management.

This key process area covers the practices for performing peer reviews. The practices identifying the specific software work products that undergo peer review are contained in the key process areas that describe the development and maintenance of each software work product.

The goals of Peer Reviews are:

1. Peer review activities are planned.
2. Defects in the software work products are identified and removed.

The top-level activities performed for Peer Reviews are:

1. Peer reviews are planned, and the plans are documented.
2. Peer reviews are performed according to a documented procedure.
3. Data on the conduct and results of the peer reviews are recorded.

Level 4: Quantitative Process Management

The purpose of Quantitative Process Management is to control the process performance of the software project quantitatively. Software process performance represents the actual results achieved from following a software process.

Quantitative Process Management involves establishing goals for the performance of the project's defined software process, which is described in the Integrated Software Management key process area, taking measurements of the process performance, analyzing these measurements, and making adjustments to maintain process performance within acceptable limits. When the process performance is stabilized within acceptable limits, the project's defined software process, the associated measurements, and the acceptable limits for the measurements are established as a baseline and used to control process performance quantitatively.

The organization collects process performance data from the software projects and uses these data to characterize the process capability (i.e., the process performance a new project can expect to attain) of the organization's standard software process, which is described in the Organization Process Definition key process area. Process capability describes the range of expected results from following a software process (i.e., the most likely outcomes that are expected from the next software project the organization undertakes). These process capability data are, in turn, used by the software projects to establish and revise their process performance goals and to analyze the performance of the projects' defined software processes.

The goals of Quantitative Process Management are:

1. The quantitative process management activities are planned.
2. The process performance of the project's defined software process is controlled quantitatively.
3. The process capability of the organization's standard software process is known in quantitative terms.

The top-level activities performed for Quantitative Process Management are:

1. The software project's plan for quantitative process management is developed according to a documented procedure.
2. The software project's quantitative process management activities are performed in accordance with the project's quantitative process management plan.
3. The strategy for the data collection and the quantitative analyses to be performed are determined based on the project's defined software process.
4. The measurement data used to control the project's defined software process quantitatively are collected according to a documented procedure.
5. The project's defined software process is analyzed and brought under quantitative control according to a documented procedure.
6. Reports documenting the results of the software project's quantitative process management activities are prepared and distributed.
7. The process capability baseline for the organization's standard software process is established and maintained according to a documented procedure.

Level 4: Software Quality Management

The purpose of Software Quality Management is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

Software Quality Management involves defining quality goals for the software products, establishing plans to achieve these goals, and monitoring and adjusting the software plans, software work products, activities, and quality goals to satisfy the needs and desires of the customer and end user for high quality products.

The practices of Software Quality Management build on the practices of the Integrated Software Management and Software Product Engineering key process areas, which establish and implement the project's defined software process, and the Quantitative Process Management key process area, which establishes a quantitative understanding of the ability of the project's defined software process to achieve the desired results.

Quantitative goals are established for the software products based on the needs of the organization, the customer, and the end users. So that these goals may be achieved, the organization establishes strategies and plans, and the project specifically adjusts its defined software process, to accomplish the quality goals.

The goals of Software Quality Management are:

1. The project's software quality management activities are planned.
2. Measurable goals for software product quality and their priorities are defined.
3. Actual progress toward achieving the quality goals for the software products is quantified and managed.

The top-level activities performed for Software Quality Management are:

1. The project's software quality plan is developed and maintained according to a documented procedure.
2. The project's software quality plan is the basis for the project's activities for software quality management.
3. The project's quantitative quality goals for the software products are defined, monitored, and revised throughout the software life cycle.
4. The quality of the project's software products is measured, analyzed, and compared to the products' quantitative quality goals on an event-driven basis.
5. The software project's quantitative quality goals for the products are allocated appropriately to the subcontractors delivering software products to the project.

Level 5: Defect Prevention

The purpose of Defect Prevention is to identify the cause of defects and prevent them from recurring.

Defect Prevention involves analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future. The defects may have been identified on other projects as well as in earlier stages or tasks of the current project. Defect prevention activities are also one mechanism for spreading lessons learned between projects.

Trends are analyzed to track the types of defects that have been encountered and to identify defects that are likely to recur. Based on an understanding of the project's defined software process and how it is implemented (as described in the Integrated Software Management and Software Product Engineering key process areas), the root causes of the defects and the implications of the defects for future activities are determined.

Both the project and the organization take specific actions to prevent recurrence of the defects. Some of the organizational actions may be handled as described in the Process Change Management key process area.

The goals of Defect Prevention are:

1. Defect prevention activities are planned.
2. Common causes of defects are sought out and identified.
3. Common causes of defects are prioritized and systematically eliminated.

The top-level activities performed for Defect Prevention are:

1. The software project develops and maintains a plan for its defect prevention activities.
2. At the beginning of a software task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities.
3. Causal analysis meetings are conducted according to a documented procedure.
4. Each of the teams assigned to coordinate defect prevention activities meets on a periodic basis to review and coordinate implementation of action proposals from the causal analysis meetings.
5. Defect prevention data are documented and tracked across the teams coordinating defect prevention activities.
6. Revisions to the organization's standard software process resulting from defect prevention actions are incorporated according to a documented procedure.
7. Revisions to the project's defined software process resulting from defect prevention actions are incorporated according to a documented procedure.
8. Members of the software engineering group and software-related groups receive feedback on the status and results of the organization's and project's defect prevention activities on a periodic basis.

Level 5: Technology Change Management

The purpose of Technology Change Management is to identify new technologies (i.e., tools, methods, and processes) and track them into the organization in an orderly manner.

Technology Change Management involves identifying, selecting, and evaluating new technologies, and incorporating effective technologies into the organization. The objective is to improve software quality, increase productivity, and decrease the cycle time for product development.

The organization establishes a group (such as a software engineering process group or a technology support group) that works with the software projects to introduce and evaluate new technologies and manage changes to existing technologies. Particular emphasis is placed on technology changes that are likely to improve the capability of the organization's standard software process (as described in the Organization Process Definition key process area).

By maintaining an awareness of software-related technology innovations and systematically evaluating and experimenting with them, the organization selects appropriate technologies to improve the quality of its software and the productivity of its software activities. Pilot efforts are performed to assess new and unproven technologies before they are incorporated into normal practice. With appropriate sponsorship of the organization's management, the selected technologies are incorporated into the organization's standard software process and current projects, as appropriate.

Changes to the organization's standard software process (as described in the Organization Process Definition key process area) and the projects' defined software processes (as described in the Integrated Software Management key process area) resulting from these technology changes are handled as described in the Process Change Management key process area.

The goals of Technology Change Management are:

1. Incorporation of technology changes are planned.
2. New technologies are evaluated to determine their effect on quality and productivity.
3. Appropriate new technologies are transferred into normal practice across the organization.

The top-level activities for Technology Change Management are:

1. The organization develops and maintains a plan for technology change management.
2. The group responsible for the organization's technology change management activities works with the software projects in identifying areas of technology change.
3. Software managers and technical staff are kept informed of new technologies.
4. The group responsible for the organization's technology change management systematically analyzes the organization's standard software process to identify areas that need or could benefit from new technology.
5. Technologies are selected and acquired for the organization and software projects according to a documented procedure.
6. Pilot efforts for improving technology are conducted, where appropriate, before a new technology is introduced into normal practice.
7. Appropriate new technologies are incorporated into the organization's standard software process according to a documented procedure.
8. Appropriate new technologies are incorporated into the projects' defined software processes according to a documented procedure.

Level 5: Process Change Management

The purpose of Process Change Management is to continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

Process Change Management involves defining process improvement goals and, with senior management sponsorship, proactively and systematically identifying, evaluating, and implementing improvements to the organization's standard software process and the projects' defined software processes on a continuous basis.

Training and incentive programs are established to enable and encourage everyone in the organization to participate in process improvement activities. Improvement opportunities are identified and evaluated for potential payback to the organization. Pilot efforts are performed to assess process changes before they are incorporated into normal practice.

When software process improvements are approved for normal practice, the organization's standard software process and the projects' defined software processes are revised as appropriate. The practices for revising the organization's standard software process are found in the Organization Process Definition key process area, and the practices for revising the projects' defined software processes are found in the Integrated Software Management key process area.

The goals of Process Change Management are:

1. Continuous process improvement is planned.
2. Participation in the organization's software process improvement activities is organization wide.
3. The organization's standard software process and the projects' defined software processes are improved continuously.

The top-level activities performed for Process Change Management are:

1. A software process improvement program is established which empowers the members of the organization to improve the processes of the organization.
2. The group responsible for the organization's software process activities (e.g., software engineering process group) coordinates the software process improvement activities.
3. The organization develops and maintains a plan for software process improvement according to a documented procedure.
4. The software process improvement activities are performed in accordance with the software process improvement plan.
5. Software process improvement proposals are handled according to a documented procedure.
6. Members of the organization actively participate in teams to develop software process improvements for assigned process areas.
7. Where appropriate, the software process improvements are installed on a pilot basis to determine their benefits and effectiveness before they are introduced into normal practice.
8. When the decision is made to transfer a software process improvement into normal practice, the improvement is implemented according to a documented procedure.
9. Records of software process improvement activities are maintained.
10. Software managers and technical staff receive feedback on the status and results of the software process improvement activities on an event-driven basis.

Abridged Practices

Appendix D: Change History

Date	Version	Change Description
1 Mar 90	0.0	Rough draft of the key practice tables; distributed to the CMM User Working Group. Version for review at the March 1990 CMM Workshop.
1 May 90	0.1	Rough draft of the key practice tables appendix to incorporate recommendations made at the March 1990 CMM Workshop against Version 0.0. Version for internal SEI peer review.
6 Jun 90	0.2	Draft, revision to the key practice tables to incorporate comments from the SEI peer review of Version 0.1. Version distributed to the CMM User Working Group for its review and comments. (Baselined version of the key practice tables).
26 Feb 91	0.3	Draft, revision to the key practice tables (Level 2 key process areas only) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board made against Version 0.2. Version for internal SEI peer review. Rough draft (first version) of overview and definitions to assist in the SEI peer review of the key practice tables, Level 2 key process areas.
18 Mar 91	0.4	Draft, revision to the key practice tables, Level 2 key process areas to incorporate comments from the SEI peer review of Version 0.3. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments. Rough draft, revision of overview and definitions to assist the CMM User Working Group and the Questionnaire Advisory Board in their review of the key practice tables, Level 2 key process areas.

Change History

10 Apr 91	0.41	Draft, revision to the key practice tables (Level 3 key process areas only, except Software Product Engineering key process area is not included) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board made against Version 0.2. Version for internal SEI peer review.
22 Apr 91	0.5	Draft, revision to the key practice tables, Level 3 key process areas (except the Software Product Engineering key process area is not included) to incorporate comments from the SEI peer review of Version 0.41. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments. Rough draft, revision of overview and definitions to assist the CMM User Working Group and the Questionnaire Advisory Board in their review of the key practice tables, Level 3 key process areas.
17 May 91	0.51	Draft, revision to the Level 2 key practice areas to incorporate comments from the CMM User Working Group made against Version 0.4 and make practices of each key process area at a single maturity level. Version for internal SEI peer review.
23 May 91	0.52	Draft, revision to the key practice tables (initial version of Software Requirements Management key process area, and consolidation of Software Requirements Analysis, Software Design, and Software Testing key process areas and addition of coding practices into the single key process area, Software Product Engineering) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board. Version for internal SEI peer review.
5 Jun 91	0.53	Preliminary baseline, revision to the Level 2 key practice areas to incorporate comments from the internal SEI peer review of Version 0.51. Version submitted to SEI's Information Management.

Change History

7 Jun 91	0.54	Draft, revision to the key practice tables (initial version of Software Project Management key process area only). Version for internal SEI peer review.
14 Jun 91	0.6	Draft, revision to the key practice tables to incorporate comments from the SEI peer review of Software Requirements Management (Version 0.52), Software Project Management (Version 0.54), and Software Product Engineering (Version 0.52) key process areas; incorporated Version 0.53 of other Level 2 key process areas. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments. Rough draft, revision of overview and definitions to assist the CMM User Working Group and the Questionnaire Advisory Board in their review of the key practice tables (Software Requirements Management, Software Project Management, and Software Product Engineering key process areas). Significant changes include (1) restructuring key process area practice tables by changing the common key features, (2) modifying the set of key process areas, and (3) rearranging the practice statements so that each key process area contains practices for a single maturity level.
21 Jun 91	0.61	Draft, revision to the key practice tables (Level 4 and 5 key process areas only) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board made against Version 0.2. Version for internal SEI peer review.
28 Jun 91	0.7	Draft, revision to the key practice tables appendix, (Level 4 and 5 key process areas only) to incorporate comments from the SEI peer review of Version 0.61. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments.

Change History

10 Jul 91	0.71	Draft, revision to the Level 3 key process areas and Software Requirements Management key process area to incorporate comments from the CMM User Working Group made against Versions 0.5 and 0.6; make practices of each key process area at a single maturity level; and renamed three key process areas (Software Requirements Management to Requirements Management, Software Project Management to Integrated Software Management, and Technical Team Coordination to Intergroup Coordination). Version for internal SEI peer review.
14 Jul 91	0.72	Draft, revision to the Level 3 key process areas and Software Requirements Management key process area to incorporate comments from the internal SEI peer review against Version 0.71. Version for internal SEI peer review.
15 Jul 91	0.73	Draft, revision to the Level 3 key process areas and Software Requirements Management key process area to incorporate comments from the internal SEI peer review.
15 Jul 91	0.74	Preliminary baseline, revision to the Level 3 key process areas and Software Requirements Management key process area (reformatting and fixing editorial defects). Version submitted to SEI's Information Management.
22 Jul 91	0.75	Draft, revision to the Level 4 and 5 key process areas to incorporate comments from the CMM User Working Group made against Version 0.7. Version for internal SEI peer review.
30 Jul 91	0.76	Preliminary baseline, revision to Level 4 and 5 key process areas to incorporate comments from internal SEI peer review. Version submitted to SEI's Information Management.

Change History

30 Jul 91	0.77	Draft, revision to the preliminary baseline of the Requirements Management key process area to incorporate comments from the 30 July 1991 Questionnaire Advisory Board meeting. Version for internal SEI peer review.
5 Aug 91	0.78	Draft, revision to Version 0.77 of the Requirements Management key process area to incorporate comments from the 31 July 1991 Questionnaire Advisory Board meeting. Version for internal SEI peer review.
7 Aug 91	0.79	Revised preliminary baseline of the Requirements Management key process area to incorporate comments from the Questionnaire Advisory Board and the internal SEI peer review against Version 0.78.
7 Aug 91	0.80	Draft, revision of overview and definitions to incorporate comments received from the CMM User Working Group and the Questionnaire Advisory Board in their review of Versions 0.4, 0.5, and 0.6 and restructure the document. Version for internal SEI peer review.
15 Aug 91	1.0	Baseline version for public release.
1 Sep 92	1.01	Draft, revision to Version 1.0 based on change requests received by 17 January 92 and recommendations from the CMM Workshop held 6-7 April 92. Training Program key process area replaced with Skills Building. Title of Technology Innovation changed to Technology Change Management.
8 Dec 92	1.02	Draft, revision to Version 1.01 to incorporate recommendations of CMM Advisory Board and comments from the internal SEI peer review. Training Program key process area restored, with minor upgrades to address skills-building concerns. Title of Process Measurement and Analysis key process area changed to Quantitative Process Management. Version for internal SEI peer review.

Change History

14 Dec 92	1.03	Draft, revision to Version 1.02 to incorporate comments from the internal SEI peer review. Title of Quality Management key process area changed to Software Quality Management. Version for internal SEI peer review.
18 Dec 92	1.04	Draft, revision to Version 1.03 to incorporate comments from the internal SEI peer review. Distributed to reviewers of version 1.01 and the CMM Advisory Board for comment and feedback.
15 Jan 93	1.05	Draft, revision to Version 1.04 to incorporate recommendations of CMM Advisory Board and comments from the internal SEI peer review. Version for internal SEI peer review.
10 Feb 93	1.1	Baseline version for public release.

Appendix E: Index

A

- Ability to Perform
 - description of O-28
- acceptance testing
 - subcontractor's products L2-55
- action item
 - data in L5-11
- action proposal
 - data in L5-11
- Activities Performed
 - description of O-28
- activity
 - definition of O-59
- allocated requirements. *See also* system requirements allocated to software
 - definition of L2-2
- assessment
 - software process L3-6
- audit
 - defect prevention L5-15
 - integrated software management L3-57
 - intergroup coordination L3-92
 - organization process definition L3-23
 - peer reviews L3-100
 - process change management L5-46
 - quantitative process management L4-16
 - requirements management L2-9
 - software baseline L2-81
 - software configuration management L2-83
 - software product engineering L3-81
 - software project planning L2-27
 - software project tracking and oversight L2-41
 - software quality management L4-32
 - software subcontract management 57
 - software work products L2-66
 - technology change management L5-29

B

- baseline
 - procedure for controlling changes to 80
- baseline configuration management
 - description of O-46
- budget

versus funding O-37

C

- Capability Maturity Model
 - background of O-7
 - description of O-7
 - evolution of O-8
 - flexibility in O-75
 - need for O-1
 - organizational structure used in O-64
 - relation to appraisal methods O-4
 - relation to documents O-62
 - relation to life cycles O-61
 - relation to technology O-62
 - sources of O-3
 - structure of O-9, O-12
 - use for small and/or commercial organizations O-2
 - uses of O-1
 - requirements for L5-7
- causal analysis meetings
 - procedure for 79
- change request
 - procedure for evaluating L2-79
- checklist for evaluating software work products L3-98
- Commitment to Perform
 - description of O-28
- commitments
 - changes to 7
 - review by senior management L2-31, L2-35
 - software project L2-12
- Common Features
 - definition of O-11, O-27
- configuration items/units
 - definition of L2-73
 - identification of L2-78
 - procedure for changing L2-79
 - recording status of L2-80
- configuration management. *See also* software configuration management and "manage and control"
 - baseline O-46
 - developmental O-46
 - versus manage and control O-46
- configuration management library system

Index

- definition of L2-73
- establishment of L2-77
- contingency planning L3-54
- contract
 - software subcontract management L2-50
- conventions
 - description of O-36
 - deviation from software development plan L2-30
- corrective actions
 - deviations from software development plan L2-30
- critical computer resources
 - basis for estimates L3-50
 - estimates for L2-23
 - management of L3-50
 - provision for reserve capacity L3-51
 - tracking of L2-36
- critical dependencies L3-89
- customer
 - examples of L2-1
 - internal versus external O-44
 - milestone review with L2-39
- review
 - software quality assurance L2-67
 - software requirements document L3-68
 - software test plan L3-75
 - software testing criteria L3-72
 - waivers from contractual software requirements L3-42
- review and approve
 - system documentation L3-77

D

- data
- data
 - defect prevention L5-11
 - evolution of practices for collecting and analyzing O-63
- defect
 - examples of root causes L5-8
 - prevention at Level 5 O-17
 - use of data on L3-78
- Defect Prevention L5-1 through L5-15
 - causal analysis meeting for L5-7
 - description of O-24
 - documentation of data on L5-11
 - examples of activities for L5-4
 - feedback on L5-12
 - kick-off meeting for task L5-6
 - management and control of data for L5-11
 - measurements for L5-13
 - plan for project L5-5
 - policy for organization L5-2
 - policy for project L5-2
 - resources and funding for L5-4
 - review
 - project manager L5-15
 - senior management L5-14
 - review and audit
 - software quality assurance L5-15
 - team for organization L5-3
 - team for project L5-3
 - tools for L5-4
 - training for L5-4
- defect prevention plan
 - peer review of L5-5
- defect prevention team
 - activities of L5-8
 - establishment of L5-3
- Defined Level
 - description of O-15
- defined software process
 - development of software design L3-69
 - development of software requirements L3-66
 - use in development of software code L3-71
- developmental configuration management
 - description of O-46
- documented procedure
 - acceptance testing of subcontractor's products L2-55
 - causal analysis meetings for defect prevention L5-7
 - changes to software subcontractor's statement of work L2-51
 - changing baselines L2-80
 - changing configuration items/units L2-79
 - control of project's defined software process L4-10
 - deriving software schedule L2-23
 - description of O-42

developing project's defined software process L3-41
 developing software configuration management plan L2-76
 development of organization's standard software process L3-15
 development of software development plan L2-18
 deviations in software engineering activities L2-67
 estimating critical computer resources L2-23
 estimating resources and costs L2-22
 estimating size of software work products L2-21
 identification of critical dependencies L3-89
 implementation of software process improvement L5-42
 incorporation of new technology into organization's standard software process L5-28
 incorporation of new technology into project's defined software process L5-28
 intergroup issues L3-90
 management of critical computer resources L3-50
 management of software costs L3-48
 management of software schedule L3-51
 milestone reviews L2-39
 monitoring subcontractor's software configuration management L2-54
 monitoring subcontractor's software quality assurance L2-53
 organization's training plan L3-30
 peer reviews L3-97
 preparation of software quality assurance plan L2-63
 process capability baseline L4-13
 products from software baseline library L2-80
 project's software development plan L3-44
 quantitative process control data L4-9
 quantitative process management plan L4-6

recording status of configuration items L2-80
 review of commitments L2-35
 reviews of software subcontractor L2-53
 revising software development plan L2-33
 revision of project's defined software process L3-43
 revisions to organization's standard software process based on defect prevention L5-12
 revisions to project's defined software process based on defect prevention L5-12
 risk management L3-52
 selection of new technologies L5-26
 selection of software subcontractor L2-49
 software baseline audits L2-81
 software process improvement plan L5-37
 software process improvement proposals L5-39
 software quality plan L4-23
 software size management L3-47
 subcontracted work L2-47
 documents
 relation to Capability Maturity Model O-62

E

effort and costs
 estimates for L2-22
 engineering groups
 dependencies between L3-89
 estimates
 critical computer resources L2-23
 effort and costs L2-22
 size of software work products L2-21

F

Fagan-style inspections
 use in Peer Reviews O-23
 first-line software manager
 definition of O-67
 review

Index

project tracking L2-38
funding
versus budget O-37

G

goals
definition of O-11
use in interpreting key practices O-75
use of O-27
group
definition of O-70
organization process focus L3-3
quantitative process management L4-4
software configuration management
L2-74
software quality assurance L2-61
technology change management L5-20
training L3-27

I

independence
need for O-73
Initial Level
description of O-13
Integrated Software Management L3-37
through L3-58
description of O-22
policy for L3-38
resources and funding for L3-39
review
project manager L3-57
senior management L3-56
training for L3-39-40
interdisciplinary engineering teams
policy for L3-84
intergroup commitments
plan for L3-88
Intergroup Coordination L3-83 through L3-92
description of O-22
measurements for L3-91
orientation in L3-86
resources and funding for L3-85
review
project manager L3-92
senior management L3-91
review and audit

software quality assurance L3-92
tools for L3-85
intergroup issues
procedure for L3-90

J

judgment
use in interpreting key practices O-75

K

key practices
conventions used in O-35
definition of O-12
description of O-26, O-31
example page O-33
implementation of O-35
intention of O-35
interpretation of O-27, O-75
structure of O-31
key process areas
categories of O-26
definition of O-10
description of O-18
description of Level 2 O-20 through O-21
description of Level 3 O-21 through O-23
description of Level 4 O-23
description of Level 5 O-23 through O-24
identification by maturity level O-19
spanning maturity levels O-24

L

leadership
description of O-36
library of software process-related
library of software process-related
documentation
definition of O-57
life cycle
relation to Capability Maturity Model
O-61

M

manage and control. *See also* software configuration management and configuration management

- defect prevention data L5-11
- noncompliance items L2-67
- process capability baseline L4-14
- process performance baseline L4-12
- quantitative process management plan L4-7
- software configuration management plan L2-77
- software development plan L2-13, L2-19, L2-34
- software planning data L2-25
- software process improvement plan L5-37
- software quality assurance plan L2-64
- software quality plan L4-25
- software replanning data L2-38
- statement of work for software project L2-15
- subcontract statement of work L2-48
- system requirements allocated to software L2-7
- versus configuration management O-46

Managed Level

- description of O-16

manager

- definition of O-64
- software quality assurance L2-62
- subcontract L2-45

managing

- versus tracking O-45

maturity level

- definition of O-10
- relation to process capability O-12
- spanning multiple O-24

maturity questionnaire

- relation to Capability Maturity Model O-4

Measurement and Analysis

- description of O-28

measurement data

- collection of L4-9
- examples of L4-9
- storage of L4-10

- techniques for analyzing L4-11
- program for organization L4-5
- measurements. *See also* Quantitative Process Management
- defect prevention L5-13
- institutionalization at Level 4 O-16
- integrated software management L3-56
- intergroup coordination L3-91
- organization process definition L3-22
- organization process focus L3-9
- peer reviews L3-99
- process change management L5-45
- quantitative process management L4-15
- requirements management L2-8
- software configuration management L2-82
- software engineering activities L3-80
- software product engineering L3-79
- software project planning L2-25
- software project tracking and oversight L2-39
- software quality assurance L2-68
- software quality management L4-31
- software subcontract management L2-55
- technology change management L5-28
- training program L3-34-35

milestones

- identification of L3-51

N

new technology

- information on L5-25
- selection of L5-26
- transfer within organization L3-8

noncompliance items

- management and control of L2-67

O

Optimizing Level

- description of O-17

Organization Process Definition L3-11

- through L3-23
- description of O-21
- measurements for L3-22
- policy for L3-12

Index

- resources and funding for L3-14
- review and audit
 - software quality assurance L3-23
- tools for L3-14
- training for L3-14
- Organization Process Focus L3-1 through L3-10
 - coordination of training for L3-8
 - description of O-21
 - group for L3-3
 - involvement of key groups L3-9
 - management sponsorship of L3-2
 - measurements for L3-9
 - orientation in L3-6
 - plan for L3-7
 - policy for L3-2
 - resources and funding for L3-4
 - review
 - senior management L3-10
 - senior management oversight of L3-3
 - tools for L3-5
 - training for L3-5
- organization's software process database L4-13
 - definition of O-56
 - examples of data in L3-47
 - storage of quantitative process management data L4-10
- organization's standard software process
 - analysis for new technology L5-25
 - analyzing capability of L4-3
 - baseline for L4-3, L4-13
 - changes to L4-15
 - configuration management of L3-17
 - coordination at organization level L3-7
 - definition of O-15, O-54
 - deviations from L3-39
 - guidelines for tailoring L3-19, O-56
 - incorporation of new technology L5-28
 - incorporation of process changes L5-43
 - peer review of L3-16
 - policy for L3-12
 - procedure for developing L3-15
 - process capability trends for L4-13
 - relation to software process definition O-50
 - revision based on defect prevention L5-12
 - tailoring of L3-41

- use in project planning and management L3-38
- waivers for deviations from L3-42
- definition of O-70
- orientation. *See also* training and required training
 - description of O-39
 - intergroup coordination L3-86
 - organization process focus L3-5
 - quantitative process management L4-6
 - software product engineering for managers L3-64
 - software product engineering for technical staff L3-64
 - training program I 3-29

P

- Pe to analysis
 - use in defect prevention L5-9
- Peer Reviews L3-93 through L3-100
 - description of O-22
 - plans for L3-97
 - policy for L3-94
 - procedure for L3-97
 - resources and funding for L3-95
 - review and audit
 - software quality assurance L3-100
 - measurements for L3-99
 - training to lead L3-95
 - training to participate in L3-96
 - defect prevention plan L5-6
 - quantitative process management plan L4-7
- peer review
 - checklist for evaluating work products L3-98
 - organization's standard software process L3-16
 - plan for organization process focus L3-7
 - software code L3-72
 - software design document L3-71
 - software life cycle L3-19
 - software process improvement plan L5-37
 - software requirements document L3-68
 - software risk management plan L3-54
 - software quality plan L4-24

- software test plan L3-74
- system documentation L3-77
- technology change management plan L5-24
 - versus review O-45
- periodic and event-driven review
 - description of O-49
- periodic review
 - description of O-48
- pilot efforts
 - technology changes L5-27
- plan
 - communication of intergroup commitments L3-88
 - defect prevention L5-5
 - description of O-40
 - formal O-41
 - informal O-41
 - organization process focus L3-7
 - peer reviews L3-97
 - process change management L5-37
 - quantitative process management L4-6
 - software configuration management L2-76
 - software development L2-14
 - software engineering L2-25
 - software quality assurance L2-63
 - software quality L4-23
 - technology change management L5-23
 - training L3-29
- plan for organization process focus
 - peer review of L3-7
- planning. *See* Software Project Planning
- policy
 - defect prevention for organization L5-2
 - defect prevention for project L5-2
 - description of O-36
 - integrated software management L3-38
 - interdisciplinary engineering teams L3-84
 - management of system requirements allocated to software L2-2
 - organization process definition L3-12
 - organization process focus L3-2
 - peer reviews L3-94
 - quantitative process management L4-2-3
 - software configuration management L2-72
- software process improvement L5-32
- software product engineering L3-60
- software project planning L2-12
- software project tracking and oversight L2-30
- software quality assurance L2-60
- software quality management L4-20
- software subcontract management L2-44
- technology change management L5-18
- training L3-26
- prerequisite items
 - description of O-39
- problem report
 - procedure for L2-79
- procedure. *See* documented procedure
- process capability
 - definition of L4-1
 - trends in L4-13
- Process Change Management L5-31 through L5-47. *See also* software process improvement
 - description of O-24
 - measurements for L5-42
 - resources and funding for L5-33
 - review
 - senior management
 - review and audit
 - software quality assurance L5-46
 - tools for L5-34
 - training for L5-34 through L5-35
- process assets
 - policy for L3-12
 - use in project planning and management L3-38
 - use in tailoring organization's standard software process L3-42
- process capability baseline L4-3
 - for organization's standard software process L4-13
 - management and control of L4-14
 - relation to maturity levels O-12
- process element
 - definition of L3-17
- process improvement. *See also* software process improvement
 - focus at Level 5 O-17
- process performance
 - definition of L4-1

Index

process performance baseline L4-13
 management and control of L4-12
 definition of L4-1
project manager
 definition of O-66
 negotiation of commitments L2-12
 review
 defect prevention L5-15
 integrated software management L3-57
 intergroup commitments L3-89
 intergroup coordination L3-92
 quantitative process management L4-16
 requirements management L2-9
 risks L2-38
 software configuration management L2-83
 software development plan L2-19
 software product engineering L3-80
 software project planning L2-26
 software project tracking and oversight L2-41
 software quality assurance L2-69
 software quality management L4-31
 software subcontract management L2-56
 statement of work L2-15
project software manager
 assignment of work tasks L2-31
 coordination of software project planning L2-15
 definition of O-67
 negotiation of commitments L2-12
 responsibilities of L2-30
 review
 project tracking L2-38
 software development plan L2-19
 statement of work L2-15
 software project planning responsibility L2-12
project tracking and oversight. *See* Software Project Tracking and Oversight
project's defined software process
 coordination at organization level L3-8
 coordination with defect prevention team L5-3
 definition of O-16, O-58
 documentation of L3-39

 guidelines for development L3-41
 incorporation of new technology L5-28
 incorporation of process changes L5-43
 integration of software engineering activities L3-65
 management and control of L3-42
 measurement and control of L4-2
 quantitative control of L4-10
 relation to quality goals L4-24
 revision based on defect prevention L5-12
 revision of L3-43
 software engineering tasks and L3-60
 use for quantitative process management L4-8
 use for software quality management L4-20
 use for software testing L3-72
 use in developing system documentation L3-76
 use in managing software project L3-44
 use of defect data L3-78
project's software development plan
 development and revision of L3-44
 definition of O-70

Q

quality function deployment (QFD) L4-23
quality goals
 allocation to subcontractors L4-30
 definition and revision of L4-27
 establishment at Level 4 O-16
quality needs
 traceability of L4-23
Quantitative Process Management L4-1
 through L4-17. *See also* measurement
 description of O-23
 group for L4-4
 measurements for L4-15
 orientation in L4-6
 plan for L4-6
 policy for L4-2-3
 reports for L4-12
 resources and funding for L4-4
 review
 project manager L4-16
 senior management L4-15
 review and audit

software quality assurance L4-16
 training for L4-6
 tools for L4-5
 quantitative control
 definition of L4-3
 quantitative process management plan
 items in L4-7

R

records
 software process improvement L5-43
 Repeatable Level
 description of O-14
 report
 quantitative process management L4-12
 software configuration management activities L2-81
 software quality assurance activities L2-67
 required training. *See also* training and orientation
 defect prevention L5-4
 description of O-38
 implementing software quality management L4-22
 leading peer reviews L3-95
 participation in peer reviews L3-96
 process change management L5-34-35
 quantitative process management L4-6
 technology change management L5-23
 Requirements Management L2-1 through L2-10
 description of O-20
 measurements for L2-8
 resources and funding for L2-5
 review
 project manager L2-9
 senior management L2-9
 review and audit
 software quality assurance L2-9
 tools for L2-5
 resources and funding
 defect prevention L5-4
 description of O-37
 integrated software management L3-39
 intergroup coordination L3-85
 organization process definition L3-14

organization process focus L3-4
 peer reviews L3-95
 process change management L5-33
 quantitative process management L4-4
 requirements management L2-5
 software configuration management L2-75
 software product engineering L3-61
 software project planning L2-16
 software project tracking and oversight L2-32
 software quality assurance L2-62
 software quality management L4-21
 software subcontract management L2-46
 technology change management L5-21
 training program L3-27
 review
 at milestones L2-39
 commitments L2-31
 defect prevention
 project manager L5-15
 senior management L5-14
 software quality assurance L5-15
 integrated software management
 project manager L3-57
 senior management L3-56
 software quality assurance L3-57
 intergroup coordination
 project manager L3-92
 senior management L3-91
 software quality assurance L3-92
 organization process definition
 software quality assurance L3-23
 organization process focus
 senior management L3-10
 peer reviews
 software quality assurance L3-100
 periodic and event-driven O-49
 periodic O-48
 process change management
 senior management L5-46
 software quality assurance L5-46
 quantitative process management
 project manager L4-16
 senior management L4-15
 software quality assurance L4-16
 requirements management
 project manager L2-9

Index

- senior management L2-9
- software quality assurance L2-9
- software configuration management
 - project manager L2-83
 - senior management L2-82
 - software quality assurance L2-83
- software product engineering
 - project manager L3-80
 - senior management L3-80
 - software quality assurance L3-81
- software project commitments L2-13, L2-17
- software project planning
 - project manager L2-26
 - senior management L2-26
 - software quality assurance L2-27
- software project tracking and oversight
 - project manager L2-41
 - senior management L2-40
 - software quality assurance L2-41
- software quality assurance
 - project manager L2-69
 - senior management L2-68
- software quality management
 - project manager L4-31
 - senior management L4-31
 - software quality assurance L4-32
- software subcontract management
 - project manager L2-56
 - senior management L2-56
 - software quality assurance L2-57
- technical aspects of intergroup coordination L3-90
- technical aspects of software subcontract L2-52
- technology change management
 - senior management L5-29
 - software quality assurance L5-29
- tracking progress L2-38
- training materials L3-33
- training program
 - senior management L3-35
- versus peer review O-45
- work products L3-90

risks

- identification of L2-24
- management of L3-52
- managing at Level 4 O-17
- tracking of L2-37

role
definition of O-64

S

senior management

- approval of tailoring organization's standard software process L3-42
- approval of waivers for organization's standard software process L3-42
- oversight of organization process focus activities L3-3
- oversight of software process improvement L5-32
- sponsorship of organization process focus L3-2

review

- commitment changes L2-7
- commitments L2-31, L2-35
- defect prevention L5-14
- integrated software management L3-56
- intergroup coordination L3-91
- organization process focus L3-10
- process change management L5-46
- quantitative process management L4-15
- requirements management L2-9
- software configuration management L2-82
- software product engineering L3-80
- software project commitments L2-13, L2-17
- software project planning L2-26
- software project tracking and oversight L2-40
- software quality assurance L2-61, L2-68
- software quality management L4-31
- software quality plan L4-25
- software subcontract management L2-56
- technology change management L5-29
- training program L3-35
- waivers from contractual software requirements L3-42

role in oversight O-48

- role in software process focus L3-3
- role in technology change management L5-19
- training for process change management L5-35
- senior manager
 - definition of O-65
- software baseline
 - audit of L2-81
- software activities and software work products
 - deviations in L2-67
- software architecture
 - development of L3-70
- software baseline library
 - check-in/out procedure L2-80
 - creating products from L2-80
 - creation of L2-80
 - definition of L2-73
- software baselines
 - audit of L2-83
 - repository for L2-77
- software code
 - configuration management of L3-72
 - development of L3-71
 - peer review of L3-72
- software commitments
 - changes to L2-30, L2-35
- software configuration control board (SCCB)
 - activities of L2-73
- software configuration management. *See also* manage and control
 - models for L3-66
 - organization's standard software process L3-17
 - software code L3-72
 - software design document L3-71
 - software requirements document L3-69
 - tools for developing and maintaining software L3-66
- Software Configuration Management L2-71 through L2-83
 - audit of L2-73
 - description of O-21
 - measurements for L2-82
 - reports of activities L2-81
 - resources and funding for L2-75
 - review
 - project manager L2-83
 - senior management L2-82
 - review and audit
 - software quality assurance L2-83
 - tools for L2-75
- software configuration management group L2-74
 - audit of software baselines L2-83
 - definition of O-72
 - monitor of subcontractor's configuration management L2-54
- software configuration management plan
 - management and control of L2-77
 - procedure for L2-76
 - use of L2-77
- software design
 - definition of L3-69
 - requirements for L3-69
- software design document L3-71
- software development L2-15
- software development plan
 - basis for tracking project L2-30
 - definition of L2-14, O-61
 - description of L2-11
 - deviations from L2-30
 - documentation of L2-31
 - for software project tracking and oversight L2-33
 - for subcontractor L2-51
 - incorporation of defect prevention activities L5-3
 - management and control of L2-13, L2-19, L2-34
 - procedure for L2-18
 - refinement of L2-39
 - relation to project's defined software process L3-37
 - revision of L2-33
 - software quality assurance group and L2-65
- software documentation
 - management and control of L3-77
- software effort and costs
 - management of L3-48
 - tracking of L2-36
- software engineering group
 - definition of O-71
 - participation in software project planning L2-17

Index

- participation on project proposal team L2-16
- review of project tracking L2-38
- review of system requirements allocated to software L2-5
- role in establishing system requirements L3-86
- training for process change management L5-34
- training for software quality management L4-22
- software engineering process group
 - coordination of software process improvement L5-36
 - definition of O-71
 - establishment of L3-4, O-15
 - relation with technology change management L5-20
 - responsibility for organization's standard software process L3-14
 - responsibility of L3-1
 - review of changes to tailoring guidelines L3-20
 - review of quantitative process management plan L4-7
 - review of tailoring organization's standard software process L3-42
- software engineering technical activities L2-37
- software life cycle
 - definition of O-55
 - documentation of L3-18
 - identification of L2-17
 - management and control of L3-19
 - peer review of L3-19
 - selection of L3-41
- software management
 - use at Level 2 O-15
- software manager
 - negotiation of commitments L2-13
 - orientation for training program L3-29
 - project tracking responsibility L2-32
 - responsibility for software work products L2-15
 - review
 - allocated requirements L2-3
 - project tracking L2-38
 - software development plan L2-19
 - statement of work L2-15
 - training for process change management L5-34
- software planning data L2-25
- software process
 - software process architecture
 - definition of L3-18, O-54
 - software process assessment (SPA) L3-6
 - software process assets
 - development of at Level 3 O-21
 - examples of O-53
 - software process capability
 - definition of O-10
 - software process database
 - coordination at organization level L3-8
 - establishing and maintaining L3-20
 - management and control of L3-21
 - use for planning and estimating L3-46
 - use of at Level 4 O-16
 - software process definition. *See also* Organization Process Focus
 - relation to Capability Maturity Model O-50
 - software process element
 - definition of O-55
 - software process focus
 - coordination of L3-7
 - software process improvement. *See also* Process Change Management and Organization Process Focus
 - feedback on L5-44
 - implementation of L5-42
 - oversight by senior management L5-32
 - pilot installation L5-42
 - plans for L5-37
 - policy for L5-32
 - program for L5-35
 - records of L5-43
 - teams for process areas L5-41
 - update of training for L5-43
 - software process improvement plan
 - items in L5-38
 - management and control of L5-37
 - peer review of L5-37
 - requirements for L5-37
 - software process improvement proposal
 - documentation of L5-10
 - procedure for handling L5-39
 - software process performance

assessment of L3-6
 definition of O-23
 software product
 definition of O-60
 software quality assurance audit of L2-66
 Software Product Engineering L3-59 through L3-82
 description of O-22
 measurements for L3-79
 orientation for managers L3-64
 orientation for technical staff L3-64
 policy for L3-60
 resources and funding for L3-61
 review
 project manager L3-80
 senior management L3-80
 review and audit
 software quality assurance L3-81
 tools for L3-61
 training for L3-63
 software project
 software project commitments
 review of L2-13, L2-17
 software project data
 recording of L2-38
 software project L2-14
 commitment for L2-12
 management of L3-44
 review of L3-55
 Software Project Planning L2-11 through L2-27
 coordination of L2-15
 description of O-20
 measurements for L2-25
 overall project planning and L2-17
 policy for L2-12
 resources and funding for L2-16
 responsibility for L2-12
 review
 project manager L2-26
 senior management L2-26
 review and audit
 software quality assurance L2-27
 tools for L2-16
 training for L2-16
 Software Project Tracking and Oversight L2-29 through L2-42
 assignment of work tasks L2-31

description of O-20
 measurements for L2-39
 policy for L2-30
 resources and funding for L2-32
 review
 project manager L2-41
 senior management L2-40
 review and audit
 software quality assurance L2-41
 software development plan for L2-33
 tools for L2-32
 training for L2-32
 Software Quality Assurance L2-59 through L2-69
 description of O-21
 manager for L2-62
 measurements for L2-68
 orientation in L2-63
 policy for L2-60
 resources and funding for L2-62
 review
 independent experts L2-69
 project manager L2-69
 senior management L2-68
 tools for L2-62
 training for L2-62
 software quality assurance group L2-61
 activities of L2-65
 audit of software products L2-66
 coordination with customer L2-67
 definition of O-72
 monitor of subcontractor's quality assurance L2-53
 reports of activities L2-67
 review
 defect prevention L5-15
 integrated software management L3-57
 intergroup coordination L3-92
 organization process definition L3-23
 peer reviews L3-100
 process change management L5-46
 quantitative process management L4-16
 requirements management L2-9
 software configuration management L2-83
 software engineering activities L2-66

Index

- software product engineering L3-81
- software project planning L2-27
- software project tracking and oversight L2-41
- software quality management L4-32
- software subcontract management L2-57
- technology change management L5-29
- software development plan and L2-65
- software quality assurance plan L2-63
 - management and control of L2-64
- software quality assurance review
 - description of O-49
- software quality goals
 - actions to resolve conflicts L4-30
 - documentation of L4-28
- Software Quality Management L4-19 through L4-32
 - description of O-23
 - measurements for L4-31
 - policy for L4-20
 - resources and funding for L4-21
 - review
 - project manager L4-31
 - senior management L4-31
 - review and audit
 - software quality assurance L4-32
 - tools for L4-21
 - training for software engineering group L4-22
 - training to implement L4-22
- software quality plan
 - documentation of quality goals L4-28
 - items in L4-25
 - management and control of L4-25
 - peer review of L4-24
 - requirements for L4-23
 - review by senior management L4-25
- software replanning data
 - management and control of L2-38
- software requirements
 - changes to L3-69
 - development of L3-66
- software requirements document
 - configuration management of L3-69
 - peer review of L3-68
- software risk
- software risk management plan
- management and control of L3-54
- peer review of L3-54
- plan for managing L3-53
- software schedule
 - management of L3-51
 - procedure for L2-23
 - tracking of L2-37
- software size
 - management of L3-47
- software subcontract
 - policy for managing L2-44
 - responsibility for managing L2-45
- Software Subcontract Management L2-43 through L2-57
 - basis for L2-50
 - contract for L2-50
 - description of O-20
 - measurements for L2-55
 - orientation in L2-46
 - responsibility for L2-46
 - review
 - project manager L2-56
 - senior management L2-56
 - review and audit
 - software quality assurance L2-57
 - training for L2-46
- software subcontractor
 - performance evaluation L2-55
 - procedure for selection of L2-49
 - procedure for reviews with L2-53
 - quality goals for L4-30
 - status reviews with L2-51
 - technical reviews with L2-52
- software subcontractor's statement of work
 - changes to L2-51
 - acceptance testing of products L2-55
- software task
 - kick-off meeting for defect prevention L5-6
- software task leader
 - definition of O-68
- software testing L3-72
- software test plan
 - peer review of L3-74
 - requirements for L3-75
- software testing criteria
 - review of L3-72
- software work product
 - audit of L2-66

- consistency of L3-78
- definition of O-60
- estimating size of L2-21
- evaluation of L3-98
- responsibility for L2-15, L2-31
- tracking size of L2-35
- software-related group
 - definition of O-71
- sponsorship
 - description of O-36
 - organization process focus L3-2
- staff
 - definition of O-68
- stage
 - definition of O-58
- standards
 - for training courses L3-33
 - software quality assurance involvement in developing L2-65
- statement of work
 - subcontract L2-47
- statement of work for software project management and control of L2-15
 - review of L2-15
- subcontract. *See* software subcontract
- subcontract manager
 - responsibilities of L2-45
- subcontract statement of work L2-47
 - management and control of L2-48
- subcontracted work
 - procedure for L2-47
 - selection of L2-47
- subpractices
 - description of O-31
- supplementary information
 - description of O-32
- system and acceptance testing L3-75
- system documentation
 - development of L3-76
 - peer review of L3-77
- system engineering group
 - definition of O-72
- system requirements
 - coordination to establish L3-86
 - responsibility for L2-3
- system requirements allocated to software. *See also* allocated requirements
 - changes to L2-7
 - definition of L2-1

- description of O-43
- documentation of L2-4
- examples of L2-4
- management and control of documentation for L3-78
- policy for managing L2-2
- review of L2-5
- system test group
 - definition of O-72

T

- Taguchi's method for robust design L4-24
- task
 - definition of O-59
- team
 - defect prevention L5-3
 - interdisciplinary engineering L3-84
- technology
- technology change
- Technology Change Management L5-17
- through L5-30
 - description of O-24
 - group for L5-20
 - measurements for L5-28
 - plan for L5-23
 - policy for L5-18
 - resources and funding for L5-21
 - review
 - senior management L5-29
 - review and audit
 - software quality assurance L5-29
 - senior management role in L5-19
 - tools for L5-21
 - training for L5-23
- technology change management plan L5-23
 - peer review of L5-24
 - data to support L5-22
 - identification of areas for L5-24
 - pilot effort for L5-27
 - relation to Capability Maturity Model O-62
- test plans
 - management and control of L3-74
- testing
 - of software L3-72
 - of system for acceptance L3-75
- tools

Index

- building and maintaining software L3-60
 - defect prevention L5-4
 - intergroup coordination L3-85
 - organization process definition L3-14
 - organization process focus L3-5
 - process change management L5-34
 - quantitative process measurement L4-5
 - requirements management L2-5
 - selection of for project L3-65
 - software configuration management L2-75
 - software product engineering L3-61
 - software project planning L2-16
 - software project tracking and oversight L2-32
 - software quality assurance L2-62
 - software quality management L4-21
 - software subcontract management L2-46
 - technology change management L5-21
 - training L3-28
 - tools for developing and maintaining software
 - configuration management of L3-66
 - tracking
 - critical computer resources L2-36
 - risks L2-37
 - size of software work products L2-35
 - Software engineering technical activities L2-37
 - software effort and costs L2-36
 - versus managing O-45
 - training. *See also* orientation and required training
 - coordination across organization L3-8
 - description of O-38
 - development of courses for L3-33
 - facilities for L3-28
 - integrated software management L3-39
 - intergroup coordination L3-85
 - organization process definition L3-14
 - organization process focus L3-5
 - plan for project L3-29, L3-30
 - policy for L3-26
 - records of L3-34
 - review of materials for L3-33
 - software configuration management L2-76
 - software product engineering L3-63
 - software project planning L2-16
 - software project tracking and oversight L2-32
 - software quality assurance L2-62
 - software subcontract management L2-46
 - tools for L3-28
 - update based on software process improvement L5-43
 - waiver for L3-34
 - training group
 - definition of O-73
 - skills and knowledge of L3-28
 - training materials
 - management and control of L3-34
 - training plan
 - items included in L3-32
 - management and control of L3-31
 - Training Program L3-25 through L3-36
 - description of O-22
 - group for L3-27
 - independent evaluation of L3-36
 - measurements for L3-34
 - orientation in L3-29
 - resources and funding for L3-27
 - review
 - senior management L3-35
 - review and audit of L3-36
 - training program elements
 - items included in L3-28
 - training vehicles
 - texamples of L3-26
- V**
- Verifying Implementation
 - description of O-28
- W**
- work products
 - review of L3-90

Change Request – Capability Maturity Model for Software, Version 1.1

Product: CMM v1.1 (including both SEI-93-TR-24 and SEI-93-TR-25)

SEI Assigned Tracking Number: _____

Name of Submitting Organization: _____

Organization Contact: _____ Telephone: _____

Mailing Address:

Date: _____ Short Title: _____

Change Location Tag: _____
(use section #, figure #, key process area ID, practice ID, etc.)

Proposed Change:

Rationale for Change:

Note: For the SEI to take appropriate action on a change request, we must have a clear description of the recommended change, along with a supporting rationale.

Send US mail to: CMM Change Requests, Software Process Program, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890.

Send packages to: CMM Change Requests, Software Process Program, Software Engineering Institute, Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA 15213-2691.

Send via Internet to: commchange@sei.cmu.edu

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS None		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-93-TR-25			5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-93-178		
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute		6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office		
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213			7b. ADDRESS (City, State and ZIP Code) ESC/AVS Hanscom Air Force Base, MA 01731		
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office		8b. OFFICE SYMBOL (if applicable) ESC/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19 628 90C 0003		
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213			10. SOURCE OF FUNDING NOS.		
			PROGRAM ELEMENT NO 63756E	PROJECT NO. N/A	TASK NO N/A
			WORK UNIT NO. N/A		
11. TITLE (Include Security Classification) Key Practices of the Capability Maturity Model, Version 1.1					
12. PERSONAL AUTHOR(S) Mark C. Paulk, Charles V. Weber, Suzanne Garcia, Mary Beth Chrissis, and Marilyn Bush					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM TO		14. DATE OF REPORT (Yr, Mo., Day) February 1993	
				15. PAGE COUNT 428 pp.	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.	capability maturity model key practices maturity model		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This document, "Key Practices of the Capability Maturity Model", provides the key practices that correspond to each maturity level of the Capability Maturity Model (CMM) and information on how to interpret the key practices. It is an elaboration of what is meant by maturity at each level of the CMM and a guide that can be used for software process improvement, software process assessments, and software capability evaluations. <div style="text-align: right;">(please turn over)</div>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPTDTIC USE			21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution		
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt. Col, USAF			22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631		22c. OFFICE SYMBOL ESC/AVS (SEI)

ABSTRACT —continued from page one, block 19

SUPPLEMENTARY

INFORMATION

ERDA AD-A 363432

Technical Report

CMU/SEI-93-TR-025

ESC-TR-93-178

February 1993

**Key Practices of the Capability Maturity ModelSM,
Version 1.1**



Mark C. Paulk

Charles V. Weber

Suzanne M. Garcia

Mary Beth Chrissis

Marilyn Bush

Unlimited distribution subject to the copyright.


Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER


Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright © 1996 by Carnegie Mellon University.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

Requests for permission to reproduce this document or to prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is <http://www.rai.com>

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145. Phone: (703) 274-7633.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.